Tampereen yliopisto

Aleksi Santahuhta

IOT-PROTOKOLLAN INTEGROINTI OPC UA:HAN KAUKOLÄMPÖJÄRJESTELMÄN TIEDONKERUUSSA

Tekniikan ja luonnontieteiden tiedekunta
Diplomityö
Joulukuu 2019

TIIVISTELMÄ

Aleksi Santahuhta: IoT-protokollan integrointi OPC UA:han kaukolämpöjärjestelmän tiedonkeruussa
Diplomityö
Tampereen yliopisto
Automaatiotekniikan diplomi-insinöörin tutkinto-ohjelma
Joulukuu 2019

Automaatiojärjestelmien tiedonsiirron tarve lisääntyy jatkuvasti. Automaatiojärjestelmät laajenevat tai niistä kerätään enemmän tietoa. Tavoitteena voi olla esimerkiksi prosessin tilannekuvan parantaminen tai prosessin optimointi. Edellä mainitut toimenpiteet lisäävät automaation tiedonsiirtoverkossa siirrettävää tietomäärää.

Automaation tiedonsiirtoverkkojen kapasiteetti ja laitteiden resurssit ovat kuitenkin rajallisia. Resurssit ovat rajallisia edullisissa tai iäkkäissä laitteissa. Tiedonsiirtoverkkojen kapasiteetti voi olla rajallinen erityisesti hajaantuneen automaation järjestelmissä, joissa laitteet ovat fyysisesti erillään ja kaukana toisistaan. Rajallisissa tiedonsiirtoverkoissa ja laitteissa protokollien on käytettävä kapasiteetti tehokkaasti. Lisäksi protokollien on oltava tietoturvallisia ja luotettavia.

Tämän työn tarkoituksena oli tutkia IoT-protokollia ja valita sopivin kaukolämpöjärjestelmän tiedonkeruuseen. Järjestelmässä käytetään palvelimien ja asiakasohjelmien väliseen tiedonsiirtoon OPC UA:ta, mutta etäasemien ja tiedonkeruujärjestelmän väliseen tiedonsiirtoon käytetään valittavaa protokollaa. Valinta tehtiin tiedonkeruujärjestelmän tiedonsiirrolle asetettujen vaatimusten perusteella. Ne pohjautuivat automaatiojärjestelmän tiedonsiirrolle asetettuihin vaatimuksiin sekä IIoT-järjestelmille asetettuihin vaatimuksiin. Vertailu suoritettiin kirjallisuustutkimuksena ja vertailussa hyödynnettiin aikaisempaa tutkimusmateriaalia sekä protokollien määrittelyjä. Tutkittavia IoT-protokollia olivat AMQP, CoAP, HTTP ja MQTT. MQTT-protokolla tuli valituksi vertailun tuloksena.

Protokollasta selvitettiin ohjelman suoritusnopeus ja muistin käyttö edullisessa ohjelmoitavassa logiikassa. Lisäksi selvitettiin protokollan sanomakoko ja kommunikoinnin käsittelynopeus. Työssä toteutettiin valitun protokollan ja OPC UA:n integraation prototyyppi sekä asiakasohjelma edulliseen ohjelmoitavaan logiikkaan. Toteutetulla järjestelmällä suoritettiin työn mittaukset. Kommunikoinnin käsittelynopeuden ja sanomakoon mittaustuloksia vertailtiin tuloksiin, jotka saatiin testissä, jossa käytettiin Siemensin ohjelmoitavissa logiikoissa käytettävää natiivia S7-kommunikointiprotokollaa.

Tutkimuksen tulosten perusteella voidaan arvioida MQTT-protokollan käytettävyyttä ja suorituskykyä edullisissa ohjelmoitavissa logiikoissa. Lisäksi työssä esitetään suosituksia integraation arkkitehtuurille sekä sanoman muodostamiseksi tiedonsiirrossa etäasemien ja tiedonkeruujärjestelmän välillä.

Avainsanat: IoT-protokolla, vertailu, suorituskyky, MQTT, integraatio, OPC UA

ABSTRACT

Aleksi Santahuhta: Integration of an IoT-protocol with OPC UA in a district heating data collection
Master of Science Thesis
Tampere University
Master's Degree Programme in Automation Engineering
December 2019

The need for data transfer is increasing constantly in industrial control systems. Industrial control systems grow or the amount of gathered data increases. The goal might be to get better image of the process or optimization of the process. Mentioned actions increase the data transferred in industrial control networks.

However, the capacity of industrial control networks and resources of devices are limited. The resources are limited in legacy or inexpensive devices. The capacity of industrial control system network might be limited especially in distributed control systems where devices are physically separate and far from each other. Protocols must use capacity efficiently in limited industrial control systems network. In addition, the protocols must be secure and reliable.

The purpose of this thesis is to study IoT-protocols and choose the most suitable protocol in the district heating data collection system. The system uses OPC UA in communication between servers and clients, but use chosen protocol in communication between data collection and substations. Choosing was based on requirements given to the communication of the data collection system. Requirements were defined based on common requirements for industrial control network and requirements defined for IIoT-systems. The comparison was done as a literature research based on previous researches and the specifications of the protocols. The studied IoT-protocols were AMQP, CoAP, HTTP and MQTT. The MQTT protocol was chosen as a result of the comparison.

A program cycle time and memory usage were studied in an inexpensive programmable logic controller when using the chosen protocol. Also, the message size and the communication handling speed of the protocol were studied. A prototype integration of OPC UA with the chosen protocol and a client program for a programmable logic controller were done. The measurements were done with the implementation. The communication handling speed were compared with results from measurements where a native Siemens S7-communication protocol were used.

The results of the study can be used to evaluate the usability and the performance of MQTT-protocol in inexpensive programmable logic controllers. In addition, the thesis makes recommendations for the architecture of the integration and for the mapping of the message in the communication between the data collection system and substations.

Keywords: IoT-protocol, comparison, performance, MQTT, integration, OPC UA

ALKUSANAT

Kiitän työn tilaajaa aiheesta ja mahdollisuudesta tehdä tämän diplomityön. Kiitos myös ajasta, jonka sain työn tekemiseen.

Kiitos työn kirjallisen osuuden ohjaajina ja työn tarkastajina toimineille professori (emeritus) Hannu Koivistolle ja professori Matti Vilkolle.

Kiitän hyvistä ideoista ja kommenteista kollegoitani Jukkaa, Aleksia ja Henriä.

Kiitos koko perheelleni tuesta työn tekemisen ajan. Erityisesti haluan kiittää vaimoani kannustuksesta ja tuesta. Lisäksi kiitän tytärtäni, joka motivoi minua tekemään työn.

Tampereella, 3.12.2019

Aleksi Santahuhta

SISÄLLYSLUETTELO

KUVALUETTELO

LYHENTEET JA MERKINNÄT

3G	Third generation, tiedonsiirtoteknologia
4G	Fourth generation, tiedonsiirtoteknologia
5G	Fifth generation, tiedonsiirtoteknologia
AMQP	Advanced Message Queuing Protocol, avoimen standardin kommunikointiprotokolla
API	Application Programmin Interface, ohjelmointirajapinta
CoAP	Constrained Application Protocol, avoimen standardin kommunikointiprotokolla
COM	Component Object Model, ohjelmistokomponenttien rajapintastandardi
DCOM	Distributed Component Object Model, Microsoft teknologia hajautettujen sovellusten kommunikointiin
DTLS	Datagram Transport Layer Security, salausprotokolla
DUP	Duplicate, kaksoiskappale
ERP	Enterprise Resource Planning, toiminnanohjausjärjestelmä
FIFO	First-In First-Out, jonotusperiaate
GPRS	General Packet Radio Service, GSM-verkossa toimiva tiedonsiirtopalvelu
HMI	Human-Machine Interface, käyttöliittymä
HTTP	Hypertext Transfer Protocol, kommunikointiprotokolla
IETF	Internet Engineering Task Force, internet-protokollien standardointiorganisaatio
ICS	Industrial Control System, automaatiojärjestelmä
IIoT	Industrial Internet of Things, Teollisuuden esineiden internet
IoT	Internet of Things, esineiden internet
IP	Internet Protocol, TCP/IP-viitemallin verkkokerroksen protokolla
ISO/IEC	International Organization for Standardization ja International Electrotechnical Commission, standardointijärjestö
JDBC	Java Database Connectivity, Javan ohjelmointirajapinta
JMS	Java Message Service, viestivälitysmalli
JSON	JavaScript Object Notation, avoimen standardin tiedostomuoto tiedonvälitykseen
M2M	Machine To Machine, konelta-koneelle
MD5	Message Digest, tiivistealgoritmi
MES	Manufacturing Execution System, tuotannonohjausjärjestelmä
MOM	Message-Oriented Middleware, viestipohjainen väliohjelmisto
MQTT	Message Queuing Telemetry Transport, avoimen standardin kommunikointiprotokolla
Node.js	Avoimen lähdekoodin alustariippumaton JavaScript-koodin suoritusympäristö
NTP	Network Time Protocol, Protokolla aikatiedon välittämiseen
OASIS	Organization for the Advancement of Structured Information Standards, standardointijärjestö
OPC DA	OPC Data Access, kommunikointiprotokolla
OPC UA	Open Platform Communication Unified Architecture, tiedonsiirtoprotokolla
PLC	Programmable Logic Controller, ohjelmoitava logiikka
POJO	Plain Old Java Object, Java olio
QoS	Quality of Service, palvelunlaatu
REST	Representational State Transfer, HTTP-protokollaan perustuva arkkitehtuurimalli
RFID	Radio Frequency Identification, radiotaajuinen etätunnistus

SASL	Simple Authentication and Security Layer, tietoturvan ja autentikoinnin viitekehys
SCADA	Supervisory Control And Data Acquisition, valvomo-ohjelmisto
SCTP	Stream Control Transmission Protocol, tietoliikenneprotokolla
SMS	Short Message Service, matkapuhelinten tekstiviestijärjestelmä
TCP	Transmission Control Protocol, tietoliikenneprotokolla
TLS	Transport Layer Security, salausprotokolla
TSN	Time-Sensitive Networking, reaaliaikaisuuden mahdollistava IEEE 802-standardin laajennus
UADP	UA Datagram Protocol, tietoliikenneprotokolla
UDP	User Datagram Protocol, tietoliikenneprotokolla
URI	Uniform Resource Identifier, tiedon tunniste tietoverkoissa
UTF-8	Unicode Transformation Format, merkkien koodausjärjestelmä
VPN	Virtual Private Network, virtuaalinen erillisverkko
XML	Extensible Markup Language, merkintäkieli

1. JOHDANTO

Automaatiojärjestelmien tiedonsiirron tarve lisääntyy jatkuvasti. Tarve lisääntyy tulevaisuudessa erityisesti IIoT-järjestelmien (engl. Industrial Internet of Things) myötä. [1] Prosessista kerätään enemmän tietoa valvomo-ohjelmistoihin, sekä tuotannon- tai toiminnanohjaukseen. Automaatiojärjestelmät laajenevat, mittauksia lisätään prosessiin, näytteenottotaajuutta halutaan kasvattaa tai laitteista halutaan enemmän tietoa. Edellä mainitut toimenpiteet kasvattavat siirrettävää tietomäärää. Tavoitteena voi olla prosessin tilannekuvan parantaminen tai halu optimoida prosessia energia- tai tuotantotehokkuuden parantamiseksi [1].

Vaikka tiedonsiirron tarve lisääntyy jatkuvasti, automaation laitteiden resurssit ja tiedonsiirtoverkkojen kapasiteetti on rajallinen ja siirrettävän tietomäärän lisäämiseen tulee haasteita. Automaation laitteilla on pitkä elinkaari ja iäkkäiden tai edullisten laitteiden resurssit voivat olla rajallisia. [2] Tiedonsiirtoyhteyden kapasiteetti voi olla erityisen rajallinen automaatiojärjestelmässä, jossa laitteita on paljon ja ne sijaitsevat fyysisesti erillään sekä kaukana toisistaan. Tällaisia järjestelmiä on jakeluteollisuudessa, joissa etäasemien tiedonsiirto on usein toteutettu hitailla langattomilla tekniikoilla, käyttäen esimerkiksi mobiiliverkkoja. Kustannussyistä näille ei toteuteta fyysistä yhteyttä. [3]

Kevyillä kommunikointiprotokollilla voidaan edesauttaa siirrettävän tiedon lisäämistä. Kommunikointiprotokollien valinta teollisuuden tiedonsiirtoon on kuitenkin haastava tehtävä. Protokollia on lukuisia ja teollisuuden tiedonsiirrolla on ominaisia, käyttötapauskohtaisia vaatimuksia, joiden käytettävien protokollien on toteutettava [4]. Protokollien valinnassa on myös huomioitava, että automaatiojärjestelmien laitteilla on pitkä elinkaari. Tästä syystä protokollan tuen on oltava toteutettavissa eri valmistajien laitteisiin, tai laitteisiin, joiden ominaisuudet ovat rajallisia. [2]

1.1 Työn tavoite

Tämän diplomityön tarkoituksena on tutkia eri IoT-protokollia (engl. Internet of Things) ja selvittää niiden soveltuvuus kaukolämpöjärjestelmän tiedonkeruuseen, jossa asiakassovelluksien ja palvelimien väliseen kommunikointiin käytetään OPC UA:ta (engl. Open Platform Communications Unified Architecture). Tiedonkeruujärjestelmällä kerätään tar-

vittavat mittaus- ja tilatiedot kaukolämpöjärjestelmän etäasemilta valvomon, raportoinnin, tuotannon- tai toiminnanohjauksen käyttöön. Työssä valitaan IoT-protokolla ja selvitetään käytettävyys tiedonkeruujärjestelmän ja etäasemien välisessä kommunikoinnissa.

Kevyen kommunikointiprotokollan käytöllä pyritään hyödyntämään tehokkaasti rajallinen tiedonsiirtoverkon ja laitteiden kapasiteetti, sekä luomaan liitettävyyttä erilaisille järjestelmän laitteille, kuten eri valmistajien logiikoille. Järjestelmien laitekannat vaihtelevat, joten kommunikointiprotokollaksi halutaan avoin ja valmistajariippumaton tietoturvallinen protokolla. Lisäksi etäluettavien asemien pitää olla helposti laajennettavissa tai lisättävissä järjestelmään.

OPC UA:ta ei käytetä kommunikointiin etäasemille, koska sen tuki puuttuu tilaajan käyttämistä edullisista S7-1200-sarjan ohjelmoitavista logiikoista sekä vanhoista logiikkamalleista. Tästä syystä IoT-protokolla, jolla kommunikoidaan järjestelmän etäasemille, integroidaan OPC UA:n kanssa. Integraatiosta toteutetaan prototyyppi, jolla saadaan kokemusta integraation arkkitehtuurista ja testataan valitun protokollan suorituskyky edullisessa ohjelmoitavassa logiikassa. IoT-protokollien vertailuista on tehty tutkimuksia [5-7], mutta tuloksista ei voida päätellä toteutettavuutta tai käytettävyyttä edullisessa ohjelmoitavassa logiikassa resurssien käytön näkökulmasta.

1.2 Tutkimuskysymykset ja menetelmät

IoT-protokollan valintaan vaikuttavat tiedonkeruujärjestelmälle asetetut vaatimukset. Vaatimukset tulivat osittain yleisistä automaatioverkon vaatimuksista sekä IIoT-toteutuksista hajautetussa järjestelmässä. IoT-protokollia vertailtiin keskenään vaatimuksiin perustuen. Vertailun tulosten perusteella valittiin testattava IoT-protokolla. Valituksi tulleelta protokollalta testattiin suorituskyky edullisessa Siemensin S7-1200-sarjan logiikassa, jotta voitiin arvioida protokollan käyttökelpoisuutta. Lisäksi protokollaa verrattiin logiikan natiiviin S7-kommunikointiprotokollaan, jotta saatiin vertailuarvot, ja voitiin paremmin arvioida protokollan suorituskykyä. Diplomityön tutkimuskysymykset ovat määritelty seuraavasti:

1. Mikä kommunikointiprotokolla täyttää järjestelmälle asetetut vaatimukset parhaiten tutkimuksen kohteena olevista protokollista?
2. Onko protokollan tuki toteutettavissa, ja minkälainen on protokollan käytettävyys S7-1200-sarjan ohjelmoitavassa logiikassa resurssien käytön näkökulmasta?
3. Minkälainen on valitun protokollan suorituskyky S7-1200-sarjan ohjelmoitavassa logiikassa ja minkälainen sen suorituskyky on verrattuna S7-protokollaan?

Ensimmäiseen tutkimuskysymykseen haettiin vastausta kirjallisuustutkimuksella. Vastatessa vaatimuksiin, hyödynnettiin vertailtaviksi valittujen protokollien määrittelyjä sekä aikaisempia tutkimuksia. Toiseen ja kolmanteen tutkimuskysymykseen vastattiin empiirisellä tutkimuksella. Kun protokollan tuki oli toteutettu ohjelmoitavaan logiikkaan, mitattiin logiikkaohjelman suoritusaika ja muistin käyttö. Lisäksi toteutettiin testijärjestelmä, jolla mitattiin IoT-protokollan ja S7-protokollan suorituskykyä. Järjestelmällä saatiin myös kokemusta jatkokehitystä varten. Mittauksien tuloksia vertailtiin keskenään.

Vertailtaviksi protokolliksi ja standardeiksi valikoitui AMQP (engl. Advanced Message Queuing Protocol), MQTT (engl. MQ Telemetry Transport), CoAP (engl. Constrained Application Protocol) ja HTTP (engl. Hypertext Transfer Protocol). AMQP, MQTT ja CoAP-protokollien valinta perustui työn tilaajan aiempaan kokemukseen kyseisistä protokollista. S7-1200-sarjan logiikoissa on Web-palvelin ominaisuus hyödyntäen HTTP-protokollaa [8]. Tästä syystä työssä tutkitaan myös HTTP-protokollaa. Protokollien valintaan vaikutti myös se, että protokollat ovat saaneet jalansijaa IoT-käytössä [9]. Tällä tavoin varmistuttiin siitä, että ohjelmistoja ja tukea on saatavilla. Jatkokehitysehdotuksissa hyödynnetään osittain OPC UA PubSub -määrittelyä, mutta sen toteuttamista kokonaisuudessaan ohjelmoitavaan logiikkaan ei tutkittu.

1.3 Työn rakenne

Kirjallisuuskatsaus alkaa luvusta 2. Siinä kerrotaan teollisuuden tietoverkosta sekä sen vaatimuksista tiedonsiirrolle ja tiedonsiirron protokollille. Lopuksi luvussa esitetään yleisesti työn aiheena olevien protokollien arkkitehtuureja ja kommunikointimalleja.

Luvussa 3 esitellään OPC UA, sen käyttökohteita, nykytila ja tulevaisuuden näkymiä. Lisäksi käydään läpi sen keskeisimmät asiat tämän työn kannalta.

Luvussa 4 käydään läpi kevyet IoT-protokollat, jotka ovat tutkimuksen kohteena. Protokollista käydään läpi keskeiset piirteet ja ominaisuudet, jotka vaikuttavat valintaan.

Luvussa 5 esitetään automaatiojärjestelmän yleiskuvaus, jonka osaksi toteutettavaa IoT-protokollan integraatiota OPC UA:han sovelletaan. Luvussa esitetään automaatiojärjestelmän asettamat vaatimukset tutkittaville protokollille, sekä vertailun tulokset. Sitten luvussa käydään läpi toteutettu testaussovellus, jossa sovelletaan kommunikointiprotokollaksi valittua IoT-protokollaa ja S7-protokollaa, sekä kuvataan suorituskykytestit.

Luvussa 6 käydään läpi tutkimuksen tulokset. Lisäksi luvussa esitetään jatkokehitystarpeita, joita testausintegraation kehityksen aikana ilmenee.

Luvussa 7 on yhteenveto työn tuloksista.

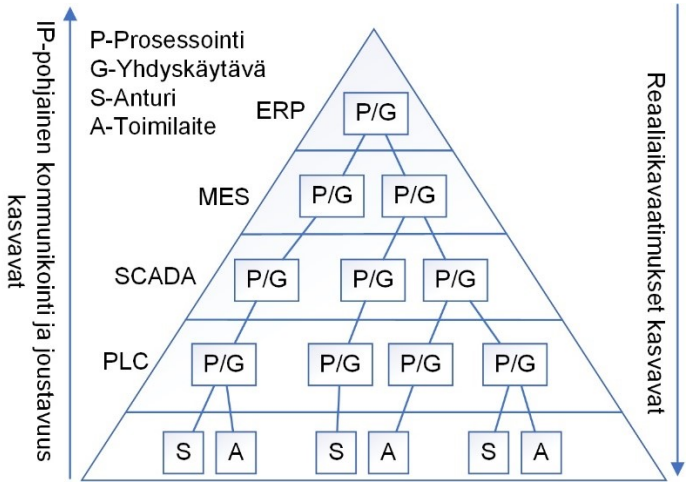
2. TEOLLISUUDEN TIEDONSIIRTO

Ensimmäisenä tässä luvussa perehdytään yleisesti teollisuuden tietoverkkojen käyttökohteisiin ja arkkitehtuurin nykytilaan. Aliluvussa 2.2 käsitellään teollisuuden esineiden internetiä ja protokollia. Siinä käydään läpi teollisuuden esineiden internetin käsite sekä sen vaikutus teollisuuden tietoverkkojen arkkitehtuuriin. Aliluvussa 2.3 käydään läpi teollisuuden internetin asettamia vaatimuksia tiedonsiirrolle ja käytettäville protokollille. Tiedonkeruujärjestelmän, jossa IoT-protokollan ja OPC UA:n integraatiota sovelletaan, tiedonsiirron vaatimukset pohjautuvat näihin vaatimuksiin. Lopuksi luvussa esitetään kommunikointiarkkitehtuureja ja -malleja, joita käytetään tutkimuksen kohteena olevissa kommunikointiprotokollissa.

2.1 Teollisuuden tietoverkot

Teollisuuden tietoverkkoja käytetään usealla teollisuudenalalla, kuten sähköntuotannossa, valmistusteollisuudessa, elintarviketeollisuudessa, vedenjakelussa, jätevedenkäsittelyssä ja kemianteollisuudessa. Melkein jokaisessa tapauksessa, jossa tarvitaan laitteiston valvontaa ja ohjausta, käytetään jonkinlaista teollisuuden tietoverkkoa. [4]

Perinteisesti teollisuuden ja automaation tietoverkkojen arkkitehtuuri esitetään ISA-95 hierarkiamallin mukaisesti [10]. Kuvassa 1 esitetään perinteinen automaation tietoverkkojen monikerroksinen arkkitehtuuri.



Kuva 1. *Perinteinen ISA-95 hierarkiamallin mukainen automaation tietoverkkojen arkkitehtuuri. Muokattu lähteestä [10].*

Kuvan 1 mukaisesti hierarkkinen malli jakaa teollisuuden tehtävät vertikaalisesti 5 eri kerrokseen. Mallin ylimmällä kerroksella on liiketoiminnan suunnittelu ja logistiikka. Näitä

hallinnoidaan yleensä ERP-järjestelmillä (engl. Enterprise Resource Planning). Toiminnanohjauksen alapuolella toteutetaan valmistuksenohjausta, eli mitä prosesseja suoritetaan ja missä järjestyksessä. Näitä toimintoja hallinnoidaan MES-järjestelmillä (engl. Manufacturing Execution Systems). Valmistuksenohjauksen alapuolella on kerros, jossa suoritetaan valvonta. Siinä laitteistoa valvotaan HMI-rajapintoja (engl. Human-Machine interface) tai SCADA-valvomojärjestelmiä (engl. Supervisory Control And Data Acquisition) käyttämällä. Laitteiston ohjaus suoritetaan valvontakerroksen alapuolella, käyttäen esimerkiksi PLC-laitteita (engl. Programmable Logic Controller) eli ohjelmoitavia logiikkoja. [11]

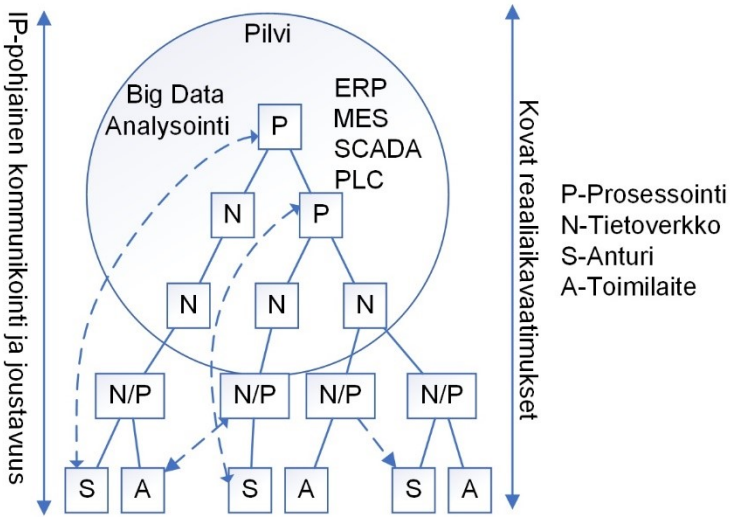
Järjestelmän arkkitehtuurin eri tasoilla käytetään yleensä eri protokollia ja tiedonsiirtomedioita. Automaation tietoverkot ovat jakautuneet ohjaus- ja laitetason tietoverkkoihin. Laitetason tietoverkkoja kutsutaan useasti kenttäväyliksi. [4] Järjestelmän tiedonsiirron reaaliaikavaatimukset kasvavat arkkitehtuurin alimmilla kerroksilla. Ylimmillä kerroksilla käytetään perinteisiä tietoverkkojen teknologioita. [10]

2.2 Teollisuuden esineiden internet

Esineiden internet -termiä (engl. Internet of Things) käytettiin ensimmäisen kerran tiettävästi vuonna 1999 Kevin Ashtonin esityksessä Procter & Gamblelle. Esitys koski ajatusta RFID:n (engl. Radio Frequency Identification) käytöstä Procter & Gamblen toimitusketjuissa. Termin käyttö yleistyi 10 vuotta sen jälkeen, kun termiä ensimmäistä kertaa käytettiin. [12] IoT:ssä internet-yhteys laajennetaan fyysisiin laitteisiin eli esineisiin. Laitteet voivat kommunikoida toistensa kanssa käyttämällä internet-yhteyttä. Laitteisiin voi olla kytketty antureita tai toimilaitteita. Laitteet voivat kerätä tai säilöä tietoa ja ne voivat tarjota tämän tiedon käytettäväksi tietoverkoissa. IoT-laitteilla ovat valmiudet kommunikoida toisten laitteiden kanssa tietoverkoissa yhdyskäytävien kautta, ilman yhdyskäytäviä tai suoraan käyttämättä tietoverkkoja. [13]

Teollisuuden esineiden internet -sovelluksissa käytetään termejä IIoT, ”teollisuuden internet” ja ”teollisuus 4.0”. Teollisuudessa IoT:lla pyritään parantamaan tuotantoa parantamalla tiedon saatavuutta sekä tuottamalla enemmän tietoa nopeammin ja tehokkaammin. [14] Teollisuuden esineiden internetillä saavutetaan hyötyä teollisuudessa. Käyttötapauksia ovat muun muassa ennakoiva huolto, inventaarion tai tuotannon optimointi. Ennakoivassa huollossa laitteiden ja koneiden tuottamaa informaatiota analysoidaan reaaliaikaisesti. Analysoidulla informaatiolla voidaan tuottaa suorituskykytietoa, jonka perusteella voidaan ennakoida huollontarpeet. Inventaariota voidaan optimoida reaaliaikaisesti, jolloin varastointia minimoidaan ja osat saadaan oikea-aikaisesti tuotantolinjoille. [15]

IIoT-järjestelmät ovat muuttamassa kuvan 1 hierarkkisen automaatiojärjestelmän rakennetta kuvan 2 mukaisiksi. Ne pyrkivät madaltamaan automaatiopyramidia sekä vähentämään kerroksia päätöksenteon ja järjestelmien ohjauksien välissä [11]. Järjestelmän eri osat ovat yhteydessä toisiinsa ilman selvää hierarkiaa. Kommunikointi pyritään toteuttamaan IP-pohjaisesti (engl. Internet Protocol) kenttälaitetasolle saakka. Tällä hetkellä se on haasteellista, koska vanhoja kenttäväyläteknologioita on vielä käytössä. [10]



Kuva 2. *Teollisuus 4.0, IIoT-arkkitehtuuri. Muokattu lähteestä [10].*

Matalampi hierarkia ja tiedon jakaminen kaikilla tasoilla vaatii suorituskykyisiä tiedonsiirtoverkkoja. Tyypilliset automaation tietoverkon vaatimukset, kuten alhainen latenssi, korkea saatavuus, aikasynkronointi ja suuret tietomäärät, ovat tuolloin vaatimuksina koko järjestelmän tiedonsiirrossa. [10]

Tiedonsiirtoverkkojen kovat vaatimukset rajoittavat IIoT-järjestelmien implementointia. IIoT-järjestelmissä latenssi saa olla korkeintaan muutaman millisekunnin. Kuitenkin tällä hetkellä esimerkiksi mobiiliverkot, jotka käyttävät 3G/4G-teknologioita (engl. third generation/ fourth generation), mahdollistavat tiedonsiirron latenssiltaan kymmeniä tai satoja millisekunteja. Lisäksi latenssien suuruus vaihtelee ja tietoverkkojen luotettavuus ei ole vaaditulla tasolla. [16,17]

Koska perinteiset teknologiat eivät täytä IIoT-järjestelmien vaatimuksia, uusia teknologioita kehitetään. Tällä hetkellä TSN-tekniikoita (engl. Time-Sensitive Networking) pidetään potentiaalisena teknologiana yhdessä OPC UA:n kanssa mahdollistamassa teollisuus 4.0. TSN-tekniikoilla voidaan tuottaa reaaliaikaiset ja deterministiset Ethernet-tietoverkot. [10] Lisäksi myös tiedonsiirtoverkkojen kehityttävä, jotta reaaliaikainen tiedonsiirto voidaan toteuttaa myös hajautetuissa ohjausjärjestelmissä. Teollisuus 4.0 toteutuminen vaatii lisäksi nopeat tiedonsiirtoyhteydet. Nopeat tiedonsiirtoyhteydet mahdollistetaan tulevaisuudessa esimerkiksi 5G-teknologioilla (engl. fifth generation). [16]

2.2.1 Kommunikointiprotokollat

Kommunikointiprotokollat määrittelevät säännöt ja käytänteet, joita käytetään kommunikointiin tietoverkoissa. Alemman tason protokollat määrittelevät yksityiskohdat esimerkiksi langattomalle tiedonsiirrolle, kun taas ylemmän tason protokollat määrittelevät esimerkiksi sanomien vaihdon sovellusten välillä. Protokolla esittelee yksityiskohtaisesti ne keinot ja tavat, joilla tiedonvaihto tapahtuu. Protokolla kuvaa muun muassa mitä tehdä virhetilanteissa tai odottamattomissa tilanteissa. Tällöin kuvataan hyväksyttävät toimenpiteet, joilla reagoidaan epätavanomaisiin tilanteisiin. [18]

Tässä tutkimuksessa keskitytään sovellusprotokolliin (engl. application protocol). Sovellusprotokollalla tarkoitetaan protokollaa, joka määrittelee, kuinka sovellukset kommunikoivat keskenään. Protokollat määrittelevät yksityiskohdat sanomien muodosta ja tarkoituksesta, joita sovellukset vaihtavat. Lisäksi protokolla määrittelee myös proseduurit, joita kommunikoinnin aikana toteutetaan. [18]

2.2.2 Protokollan valinta

Esineiden internetissä laitteet kommunikoivat hyvin erilaisia tiedonsiirtoyhteyksiä käyttäen, erilaisissa ympäristöissä ja arkkitehtuureissa. Tästä syystä on oltava useita eri käyttötapauksiin soveltuvia protokollia. [19] Esineiden internetissä yleisesti käytettyjä protokollia kutsutaan IoT- tai IIoT-protokolliksi [7,19,20].

Standardoidun ja tehokkaan protokollan valinta on haastava tehtävä, koska valinta riippuu järjestelmän luonteesta ja sen viestinnän vaatimuksista. On kehitetty useita protokollia, ja niitä on käyttöönotettu useissa eri organisaatioissa pohjautuen käyttöönottavan järjestelmän vaatimuksiin. Yksikään protokolla ei täytä kaikkia asetettuja vaatimuksia kaikissa eri tyyppisissä järjestelmissä. Kommunikointiprotokollan valinta on toistuva ongelma. Tästä syystä on tärkeää tuntea eri protokollien hyvät ja huonot puolet laajasti hyväksytyistä ja käyttöönotetuista protokollista, jotta voidaan määritellä niille parhaiten soveltuvat käyttötapaukset. [5]

2.3 Teollisuuden tietoverkon vaatimukset ja suunnittelunäkökohdat

Tässä aliluvussa käsitellään teollisuuden tietoverkon vaatimuksia. Lisäksi esitetään huomioon otettavia suunnittelunäkökulmia, joita esiintyy implementoitaessa IIoT:ssä käytettyjä ratkaisuja. Jokainen teollisuudenala asettaa omat hieman eriävät, mutta yleisesti samankaltaiset vaatimukset, jotka voidaan jaotella seuraaviin ryhmiin: valmistava teolli-

suus, prosessinohjaus, rakennusautomaatio, jakeluteollisuus, logistiikka ja sulautetut järjestelmät. [4] Tässä tutkimuksessa perehdytään tiedonkeruujärjestelmän tiedonsiirtoon kaukolämpöjärjestelmissä, eli jakeluteollisuuden tietoverkkoihin.

Toimialan ja arkkitehtuurin asettamat vaatimukset

Jakeluteollisuuden tietoverkot muistuttavat tavallisesti valmistavan teollisuuden tietoverkkoja, vaikka yleensä ohjattavat laitteet ovat yhteydessä toisiinsa. Valmistavassa teollisuudessa prosessit jaetaan soluihin, jotka ovat yleisesti autonomisia pienellä alueella. Järjestelmän osat ovat yhteydessä hierarkian ylimmillä tasoilla, kuten tehtaan ohjausjärjestelmässä. Jakeluteollisuuden järjestelmä kattaa suuren fyysisen alueen ja etäisyydet. Tämä aiheuttaa haasteita tietoverkkojen yhdistettävyydelle, mutta toisaalta kasvattaa aikaa, joilla prosessin osat vaikuttavat toisiinsa. [4] Lisäksi jakeluteollisuudessa käytetään useita eri tiedonsiirtoteknologioita. Tietoa voidaan siirtää kuituyhteyksillä, puhelinverkossa tai mobiiliverkoissa. Tällöin myös tietoverkkojen kapasiteetti vaihtelee suuresti. [3]

Sovellettaessa teollisuuden esineiden internetiä, arkkitehtuurit voivat rakentua teollisuuden toimialan mukaisesti. Toimiala ja arkkitehtuuri asettavat omat vaatimukset tietoverkoille. Jotkut järjestelmät vaativat luotettavaa alhaisen latenssin kommunikointia, kun taas toisissa järjestelmissä on otettava huomioon alhainen virrankulutus ja pitkän kantaman langaton tiedonsiirto. [2] Järjestelmäarkkitehtuuria suunniteltaessa on mietittävä, esimerkiksi toteutetaanko kommunikointi laitteelta-laitteelle vai käyttäen välittäjää. Laitteelta-laitteelle kommunikoinnilla voidaan saavuttaa esimerkiksi alhaisempi latenssi verrattuna järjestelmään, jossa käytetään välittäjää. [21] Kommunikointiarkkitehtuureista kerrotaan tarkemmin aliluvussa 2.4.

Determinismi

Deterministisessä tietoverkossa tieto täytyy siirtää siten, että tiedon siirtämiseen käytetty aika on ennustettavissa. Tiedon siirtämisessä kuluvaan aikaan ei muodostu suurta vaihtelua. Tuolloin signaalin latenssi on tietyn suuruinen. [4]

Deterministisyys on hyvin ominainen vaatimus teollisuuden tietoverkolta. Monet teollisuuden toteutukset vaativat determinististä kommunikointia. Teollisuuden automaatio, esimerkiksi liikkeenohjaus, vaatii determinististä erittäin alhaisen latenssin tiedonsiirtoa. Suurella latenssin vaihtelulla on myös negatiivinen vaikutus säätöpiireihin. [4] Näissä järjestelmissä käytetään protokollia ja teknologioita, jotka mahdollistavat vain pienen vaihtelun tiedonsiirron välitysajoissa. Näin varmistetaan sovellusten tiedon eheys ja ennustettava järjestelmän suorituskyky. [2]

Reaaliaikavaatimukset

Teollisuuden tietoverkot ovat usein aikakriittisiä, ja kriteereinä ovat sovelluskohtaiset hyväksyttävät viiveen sekä latenssin vaihtelun tasot [22]. Nopeudet, joilla prosessit ja laitteistot toimivat vaativat tiedon välityksen ja käsittelyn tapahtuvan niin nopeasti kuin mahdollista. Yleissääntö on, että tietoliikenteen vasteaika on pienempi kuin mittaamisen näytteenottojen välinen aika. Tiedonsiirron viiveet vaikuttavat säätöpiirien suorituskykyyn, erityisesti suljettuihin säätöpiireihin. Reaaliaikavaatimuksia pyritään lieventämään asteittain mitä korkeammalle tietoverkon hierarkiassa noustaan. [4]

Tietopakettien koko

Teollisuuden tietoverkkojen tietopakettien koot ovat melko pieniä arkkitehtuurien alimmilla tasoilla. Tietopaketit sisältävät yleensä erillisiä mittaus-, tila- tai ohjaustietoja sanoman otsikkokenttien lisäksi. Tietopaketit ovat tuolloin muutaman tavun kokoisia, jos kuljetettavana on esimerkiksi kahden tai neljän tavun mittainen mittaustieto tai yksittäinen bitin suuruinen tilatieto. Teollisuuden tietoverkot vaativat protokollia, jotka keskittyvät siirtämään tehokkaasti pieniä tietopaketteja. [4] Lisäksi automaatioverkoissa, joissa laitteet ovat fyysisesti erillään ja laajalla alueella, kuten sähkön- tai vedenjakelussa, joudutaan tiedonsiirto toteuttamaan useasti langattomilla tekniikoilla. Langattomien tiedonsiirtoverkkojen tiedonsiirtonopeudet voivat olla alhaiset, jolloin kannattaa suosia pieniä siirrettäviä tietomääriä. [22]

Jaksollinen ja jaksoton tiedonsiirto

Teollisuuden tietoverkoissa siirretään jaksollista tietoa, joissa näytteenotto suoritetaan tasaisin väliajoin. Näytteenottovälit riippuvat ohjausjärjestelmän vaatimuksista, jolloin ne voivat vaihdella esimerkiksi laitekohtaisesti. Lisäksi tietoverkoissa siirretään tapahtumapohjaista jaksotonta tietoa, kuten tilojen muutokset tai hälytykset. [4]

Järjestys ja eheyden ylläpito

Teollisuuden tietoverkoissa, erityisesti jaksottomassa tiedonsiirrossa, täytyy tietää tapahtumien ajankohta tai tapahtumajärjestys. Näissä järjestelmissä käytetään aikaleimausta tai synkronoituja kelloja. Tietoliikenneprotokolliin ei ole tavallisesti toteutettu järjestyksen tai ajallisen eheyden varmistusta, kuten TCP/IP-protokollassa (engl. Transmission Control Protocol/Internet Protocol). [4]

Yhteensopivuus

Päästä-päähän-yhteensopivuus on IIoT-järjestelmien haaste, koska järjestelmissä on paljon eri alustoilla olevia laitteita. Sovellussuunnittelijoiden ja laitevalmistajien tulisi ottaa huomioon yhteensopivuus. [23] Käytöillä, antureilla, ohjelmoitavilla logiikoilla ja muilla

automaatiolaitteilla on pitkä elinkaari. Jotta IIoT:tä voidaan hyödyntää automaatiojärjestelmässä, täytyy varmistua siitä, että laitteet pystyvät kommunikoimaan keskenään. [24]

Luotettavuus

Luotettavuudella halutaan varmistaa osaltaan järjestelmältä sen määritelmän mukainen toiminta. Sillä pyritään varmistamaan järjestelmän viestien kuljetus ja sillä on läheinen suhde tiedon saavutettavuuden kanssa, sillä luotettavuudella taataan tiedon ja palveluiden saavutettavuus. [23] Vaadittavan luotettavuuden tason määrittelee käytettävä sovellus [21]. Luotettavuus on kriittistä ja sille on tiukat vaatimukset järjestelmissä, joissa toteutetaan toimintoja hätätilanteissa. Sitä täytyy implementoida ohjelmiston ja laitteiston kaikilla tasoilla. Jotta saavutetaan tehokkuutta, taustalla olevan kommunikaation täytyy olla luotettavaa, koska epäluotettava tilannekuva, tiedon keräys, prosessointi ja tiedonsiirto voivat johtaa pitkiin viiveisiin, tiedon menettämiseen ja lopuksi vääriin päätöksiin. Luotettavuutta voidaan lisätä esimerkiksi mekanismeilla, joilla minimoidaan sanomien katoaminen. [23]

Standardien avoimuus ja standardoinnin taso

Hyvin standardoidut ja avoimet teknologiat sekä protokollat mahdollistavat teollisuudessa eri toimittajien ja järjestelmien yhteensopivuuden [2]. Käyttäjät eivät hyväksy järjestelmiä, joissa ollaan sidottuja tiettyihin laitetoimittajiin tai suljettuihin arkkitehtuureihin. Avoimuus voi myös lisätä kilpailua toimittajien kesken, josta voi seurata innovaatioita, tehokkuutta, laatua tai kustannusten kasvun rajoittumista. Avoimet kommunikointiprotokollat mahdollistavat protokollien vertailun, koska protokollien määrittely on saatavissa. [25]

Skaalautuvuus

Skaalautuvuus on ominainen IIoT-järjestelmän vaatimus. Useasti IIoT-järjestelmän käyttö alkaa pienestä osasta, jotta siihen ei sitoudu pääomaa. Järjestelmä muuttuu tai laajenee ja tästä syystä tietoverkon ja protokollien täytyy kyetä toimimaan laajentuvassa järjestelmässä. [26,27] Uusien menetelmien lisääminen ja uusien laitteistojen tukeminen ei ole helppoa ympäristössä, jossa on monipuolisesti eri alustoja ja kommunikointiprotokollia [23].

Tietoturva

Teollisuuden tietoverkoissa esiintyy samoja tietoturvauhkia kuin tavallisissa tietoverkoissa [4]. Lisäksi kriittisen infrastruktuurin, esimerkiksi ydinvoimalan tai jakeluteollisuuden tietoverkoissa, on suurempi kyberterrorismin uhka, koska näillä järjestelmillä on

suuri vaikutus kansalliseen turvallisuuteen. [28] Teollisuuden tietoverkkojen lisävaatimuksien ja suunnittelunäkökulmien takia tietoturvallisuuden implementointi voi olla haastavampaa kuin tavallisiin tietoverkkoihin. Tietoverkon tietoturvalla halutaan toteuttaa tiedon luottamuksellisuutta, eheyttä, saatavuutta, autentikointia, auktorisointia, kiistämättömyyttä ja suojaa kolmansilta osapuolilta. Edellä mainittujen ominaisuuksien puute voi johtaa tietoverkon vikaantumiseen ja tällä voi olla vakavat seuraukset. [4]

Kun digitaalisia automaatiojärjestelmiä kehitettiin ja ensimmäisiä otettiin käyttöön, ohjausjärjestelmien tietoverkot olivat fyysisesti eristettyjä muista järjestelmistä. Käytössä oleva teknologia oli harvoin yhteydessä teollisuusympäristön ulkopuolelle. Tuolloin pääuhat järjestelmän eheydelle olivat tapaturmaiset häiriöt tai tyytymättömien työntekijöiden tahalliset teot. [4]

Koska ohjausjärjestelmien luonne on muuttunut, tietoturvallisuuden tilanne on myös muuttunut suuresti. Ohjaimet perustuvat tietokoneisiin, laitteistot ovat verkoissa ja mahdollisesti laitteisiin on saatavilla yhteys myös internetistä. Kun Ethernet-tekniikoista tuli hierarkian korkeimpien tasojen pääteknologioita ja ulkoisten yhteyksien määrä teollisuuden tietoverkkoon kasvoi, tietoturvan tarve tunnistettiin. [4]

Tietoturvaa pidetään myös yhtenä esineiden internetin suurimpana huolenaiheena [29]. Tietoverkoissa siirretään yksityistä tietoa ja tiedolla ohjataan mahdollisesti vaarallisia fyysisiä järjestelmiä [25]. Koska järjestelmän tiedot kulkevat internetissä useiden reitittimien ja kytkimien läpi, tarvitaan kunnollisia salausmekanismeja, jotta tiedon luottamuksellisuus voidaan taata [30].

2.4 Kommunikointiarkkitehtuurit

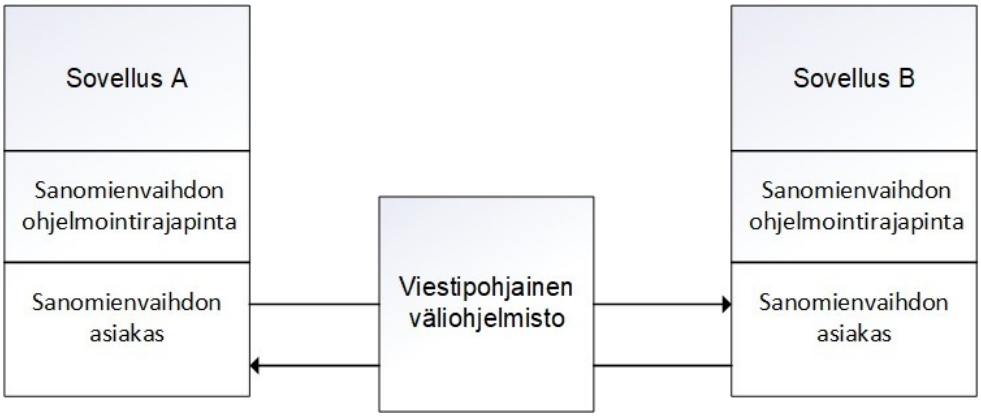
Tässä aliluvussa esitellään tutkimuksen kohteena olevien kommunikointiprotokollien käyttämiä kommunikointimalleja. Aliluvussa 2.4.1 esitetään sanomajono ja julkaise-tilaamallit, jotka käyttävät viestipohjaista väliohjelmistoa. Aliluvussa 2.4.2 esitellään asiakaspalvelinpohjainen kommunikointi.

2.4.1 Viestipohjainen väliohjelmisto

Modernit järjestelmien käyttöympäristöt ovat kompleksisia. Niissä käytetään useita ohjelmointikieliä, laitealustoja ja käyttöjärjestelmiä. Järjestelmien vaatimuksina ovat muokattavuus, korkea luotettavuus, hyvä suorituskyky ja tietoturva. Näiden lisäksi on ylläpidettävä korkeat palvelunlaatutasot. Näissä ympäristöissä suorat etäproseduurikutsuihin perustuvat mekanismit epäonnistuvat. [31]

Näitä järjestelmiä varten on kehitetty viestipohjaiset väliohjelmistot, MOM, (engl. Message-Oriented Middleware), joilla toteutetaan kommunikointi hajautetun järjestelmän eri osien välillä. Viestipohjaisen väliohjelmiston asiakas lähettää ja vastaanottaa toisten asiakkaiden sanomia käyttämällä viestin välittämiseen palvelinta. Viestipohjainen väliohjelmisto mahdollistaa joustavan ja yhtenäisen järjestelmän luonnin, jossa voidaan muokata osia järjestelmästä ilman, että tarvitsee muokata muuta osaa järjestelmästä. [31]

Kuva 3 esittää arkkitehtuuria, jossa käytetään viestipohjaista väliohjelmistoa. Asiakassovellukset käyttävät yksinkertaisia ohjelmointirajapintoja sanomien muodostamiseen. Tämän jälkeen sanomat lähetetään viestipohjaiselle väliohjelmistolle välitettäväksi yhdelle tai usealle vastaanottajalle. [32]



Kuva 3. *Viestipohjainen väliohjelmisto. Muokattu lähteestä [32].*

Viestipohjaisen väliohjelmistolla saavutettaviksi ominaisuuksiksi luetaan muun muassa lähettäjien ja vastaanottajien irti kytkentä, skaalautuvuus ja saatavuus [31]. Viestipohjainen väliohjelmisto on luonnostaan asynkroninen teknologia, jossa lähettäjät ja vastaanottajat eivät ole tiukasti kytkettyjä, toisin kuin synkroniset väliohjelmistoteknologiat. Irti kytkevä infrastruktuuri toteutetaan käyttämällä välissä olevaa viestijonoa. Lähettäjä lähettää viestin väliohjelmistolle, joka sijoittaa sanoman viestijonoon. Lähettäjät eivät ole tietoisia, mikä sovellus lopulta käyttää sanomaa. [33]

Asynkronisen luonteen sekä lähettäjien ja vastaanottajien löyhän liitoksen myötä lähettäjät eivät tarvitse vastausta sanomiin. Lähettäjä voi siirtyä tehtävässään eteenpäin heti lähetyksen jälkeen, tai lähettäjä ei tarvitse välitöntä vastausta sanomaan tai pyyntöön. Vastaanottajalla voi kulua aikaa pyynnön käsittelyyn ja tällä välin lähettäjä voi jatkaa tehtäviään. Lisäksi tietoliikenneyhteyden ei tarvitse olla jatkuvasti aktiivinen välittäjän sekä lähettäjien tai vastaanottajien välillä. Lähettäjä luottaa väliohjelmiston kykyyn toimittaa sanomat tai säilöä sanomat siihen asti, kunnes yhteydet palautuvat. [33]

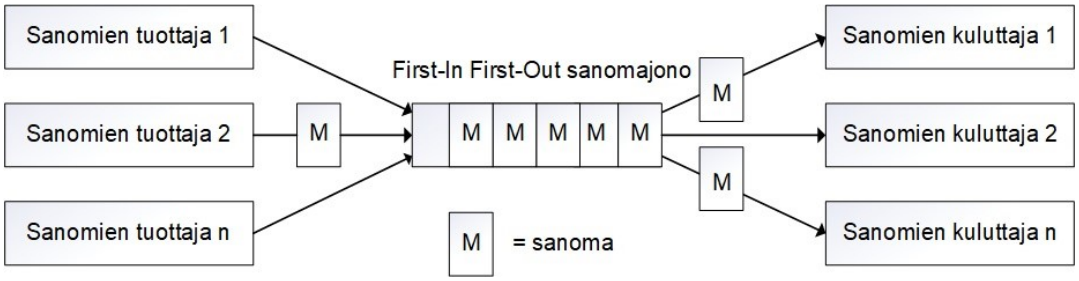
Viestipohjainen väliohjelmisto irti kytkee myös siihen liitettyjen osajärjestelmien ominaisuudet. Alajärjestelmät voidaan skaalata itsenäisesti ilman, että sillä olisi vaikutusta tai

vaikuttaa vain lievästi muihin järjestelmän osiin. Väliohjelmistot sisältävät useasti ruuhkanhallintaan liittyviä ominaisuuksia, jolloin voidaan hallita osajärjestelmissä tapahtuvia piikkejä aktiviteeteissa. [31]

Viestipohjaisilla väliohjelmilla voidaan toteuttaa korkea saavutettavuus ja mahdollistetaan järjestelmien jatkuva toiminta sekä katkoksien sujuva hallinta. Väliohjelmat eivät vaadi jokaisen osajärjestelmän jatkuvaa samanaikaista saatavuutta. Osajärjestelmien virhe- tai vikatilanteet eivät lamaannuta koko järjestelmää. [31]

Sanomajono

Sanomajonot ovat tärkeä konsepti viestipohjaisissa väliohjelmissa. Jonot mahdollistavat sanomien säilönnän. Sovellusasiakkaat voivat lähettää sanomia jonoon tai vastaanottaa sanomia jonosta, jolloin jonot ovat keskeisessä osassa asynkronisessa kommunikoinnissa. Kuva 4 esittää sanomajonoa, joka vastaanottaa sanomat ja lähettää ne eteenpäin. Perinteinen sanomajono viestipohjaisissa väliohjelmissa toimii FIFO-periaatteella (engl. First-In First-Out), jossa ensimmäinen vastaanotettu sanoma lähetetään jonosta ensimmäisenä. [31]



Kuva 4. *Sanomajono, jossa tuottajat lähettävät jonoon sanomia ja kuluttajat saavat sanomat jonosta. Muokattu lähteestä [31].*

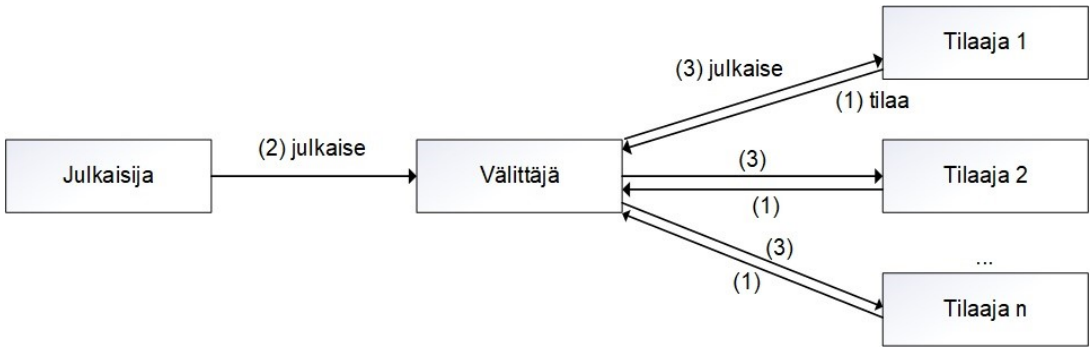
Viestipohjaisia väliohjelmistoja implementoidaan yleensä siten, että palvelin käsittelee usean rinnakkaisen asiakkaan sanomia. Palvelin luo ja ylläpitää useita sanomajonoja ja pystyy käsittelemään useita samanaikaisesti saapuvia sanomia. [33] Sanomajonoihin voidaan konfiguroida useita eri ominaisuuksia. Näitä voivat olla muun muassa jonon nimi, koko, tallennuksien alaraja tai sanomien järjestelyalgoritmi. Mobiiliasiakkaille tai asiakkaille, joilla ei ole jatkuvaa yhteyttä esimerkiksi epävakaiden yhteyksien takia, jonotuksesta on erityistä hyötyä. Jonosta voidaan tehdä tietovaranto, josta voidaan jaksottaisesti saada uudet sanomat. [31]

Julkaise-tilaa

Julkaise-tilaa-kommunikointimallin pääentiteetit ovat sisällön julkaisijat ja tilaajat. Julkaisija havaitsee tapahtuman ja julkaisee sen ilmoituksena. Ilmoitukseen on kapseloitu tapahtumaan kuuluva tieto. Tapahtumat voidaan lajitella esimerkiksi niiden ominaisuuksien mukaisesti. [34]

Ilmoitusta kutsutaan julkaisuksi. Kun ilmoitus on julkaistu, järjestelmän täytyy lähettää sanoma siitä kiinnostuneille vastaanottajille eli tilaajille. Tilaaja on kokonaisuus, joka on ilmaissut kiinnostuksen tiettyihin tapahtumiin. [34]

Julkaisijoiden suora ilmoitusten lähetys tilaajille ei ole skaalautuvaa. Tästä syystä on kehitetty tekniikoita hajautettuun ilmoitusprosessiin. Tapahtumavälittäjä tai reititin on julkaise-tilaa-mallin komponentti, joka välittää ilmoitukset tietoverkossa. [34] Kuva 5 esittää Julkaise-tilaa-kommunikointimallia, jossa julkaisijan ilmoitus lähetetään välittäjälle. Välittäjä lähettää edelleen sanomat kaikille niille tilaajille, jotka ovat kiinnostuneita kyseisestä ilmoituksesta. [19]



Kuva 5. *Julkaise-tilaa-kommunikointimalli. Muokattu lähteestä [19].*

2.4.2 Asiakas-palvelin

Asiakas-palvelin-mallilla tarkoitetaan järjestelmää, jossa tehtävät ovat hajautettu kahden eri tyyppisen kokonaisuuden välille: asiakkaiden ja palvelimien. Asiakas on mikä tahansa sovellus, joka tuottaa palvelupyyntöjä palvelimelle. Palvelin on sovellus, joka tuottaa pyydetyt palvelut asiakkaalle. Palvelimet voivat tuottaa palveluita useammalle kuin yhdelle asiakkaalle ja asiakkaat voivat pyytää palveluita useammalta verkon palvelimelta. Lisäksi palvelimet voivat toimia asiakkaina muille palvelimille. [35]

Arkkitehtuurin hyötyjä ovat muun muassa kuormituksen hajauttaminen sekä järjestelmien yhteensopivuus ja skaalautuvuus. Asiakkaan ei tarvitse tuntea palvelimen tuottamien resurssien tai palveluiden toteutustapaa. Tuolloin on mahdollista muokata palvelinta ilman, että asiakassovellusta muokataan. Lisäksi tuolloin mahdollistetaan erilaiset alustat ja asiakassovellukset. [35]

3. OPC UNIFIED ARCHITECTURE

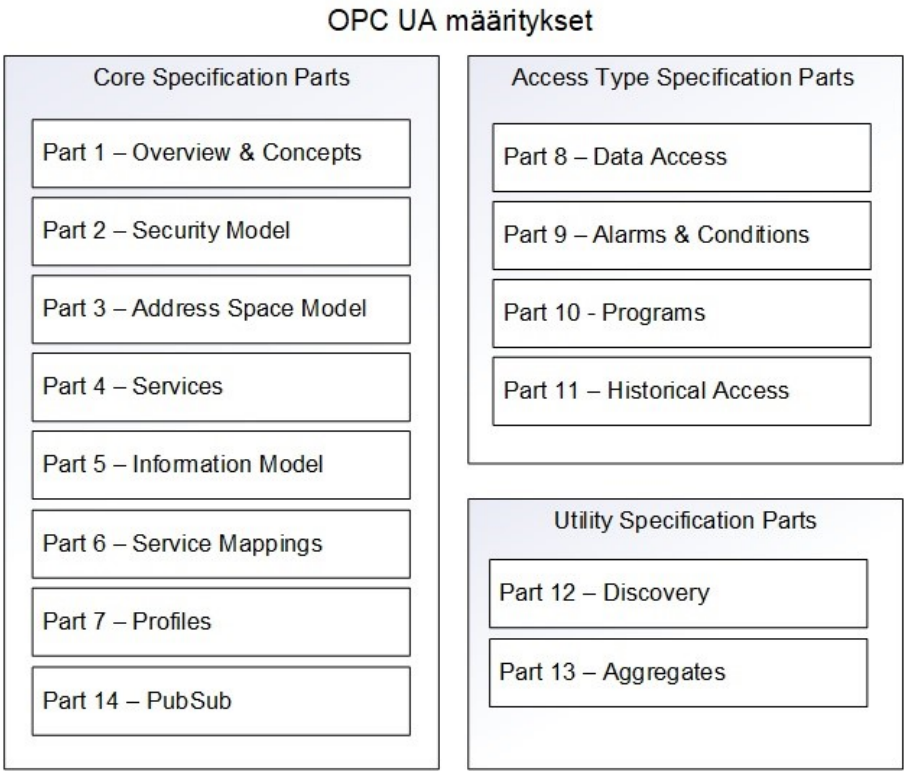
Tässä luvussa esitellään OPC UA -kommunikointiprotokolla. Luvussa esitetään OPC UA:n käyttökohteita, levinneisyyttä ja tulevaisuuden näkymiä. Lisäksi luvussa käydään läpi OPC UA:n keskeisimpiä käsitteitä tämän työn kannalta.

OPC UA on teollisuusautomaation avoin ja alustariippumaton kommunikointiprotokolla, jonka on kehittänyt OPC Foundation [36]. Ensimmäinen ja laajasti käytetty OPC Classic -standardi, OPC DA (engl. OPC Data Access), kehitettiin kommunikointiajurien rajapinnaksi. Tällä mahdollistettiin standardoitu tapa kirjoittaa ja lukea automaatiolaitteiden tietoa. Yleisimmät käyttökohteet olivat HMI, eli käyttöliittymissä ja SCADA-valvomojärjestelmissä, joissa OPC:tä käytetään tiedon lukemiseen eri laitetoimittajien automaatiolaitteista. [37]

OPC UA:n edeltäjän OPC:n rajapinnat perustuvat Microsoftin COM- (engl. Component Object Model) ja DCOM-teknologioihin (engl. Distributed Component Object Model). Teknologioiden hyödyt olivat rajapintojen valmis määrittely, jolloin OPC-määrittelyyn ei ollut tarvetta kuvata erillisiä tietoverkkoprotokollia tai kommunikointimekanismeja. Teknologian huonoina puolina olivat OPC:n Windows-alustariippuvuus ja DCOM-teknologian haasteet, kuten konfiguroinnin vaikeus. [37]

OPC UA kehitettiin, koska OPC-jäsenyritykset halusivat eroon COM-pohjaisista ratkaisuista menettämättä suorituskykyä tai ominaisuuksia. Lisäksi haluttiin kehittää täysin alustariippumattoman, laajennettavan ja monipuolisen mallinnuksen, jolla voidaan kuvata monimutkaisia järjestelmiä. OPC UA:n vaatimukset jakautuvat hajautettujen järjestelmien kommunikointiin ja tiedon mallinnukseen. OPC UA:lle asetettuja vaatimuksia ovat luotettavuus, laajennettavuus, yhteensopivuus, tietoturva ja alustariippumattomuus. [37]

OPC UA standardista pyritään julkaisemaan päivitetty versio 3 vuoden välein. Uusin versio on 1.04 on julkaistu 2018. OPC UA koostuu 14 osasta, jotka esitetään kuvassa 6. Viimeisin lisätty osa OPC UA:n määrittelyyn on PubSub. [38] Sen käyttökohteista, kehityssuunnista, tulevaisuuden näkymästä ja rakenteesta kerrotaan myöhemmin tämän luvun loppupuolella.



Kuva 6. *OPC UA:n määritykset. Muokattu lähteestä [39].*

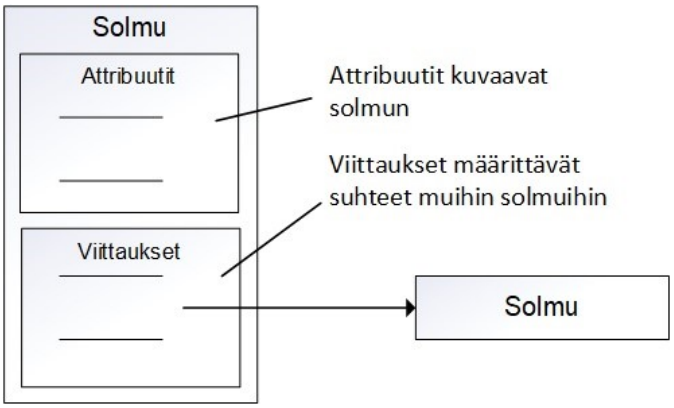
OPC UA:n määrittely tunnetaan laajasti teollisuusautomaatiossa. Sitä käytetään järjestelmien yhteensovituksessa sekä tiedonvaihdon toteutuksessa automaation hierarkian alimmilta tasoilta ylimmille tasoille. OPC UA pyrkii toteuttamaan tiedonkulun vertikaalisen ja horisontaalisen integraation. [40] OPC UA:n tuki on saatavilla ja yleistyy myös ohjelmoitavissa logiikoissa [41,42]. Lisäksi OPC UA ja TSN-tekniikoita pidetään teollisuus 4.0 mahdollistavina tekniikoina ja näitä tekniikoita pyritään hyödyntämään tulevaisuudessa myös järjestelmissä, joissa ohjaus toteutetaan virtuaalisilla ohjelmoitavilla logiikoilla [10].

OPC UA:ta käytetään lisäksi teollisuusautomaation ulkopuolella. Aluksi OPC UA keskittyi ainoastaan teollisuusautomaation tarpeisiin, siitä tuli nopeasti sovelias ratkaisu yhdistettävyyden toteuttamiseen myös teollisuuden ulkopuolella. Käytettävyys teollisuuden ulkopuolella johtuu siitä, että OPC UA on itsenäinen, eikä toimittaja- tai käyttöjärjestelmäsidonnainen. [40]

Solmut ja viittaukset

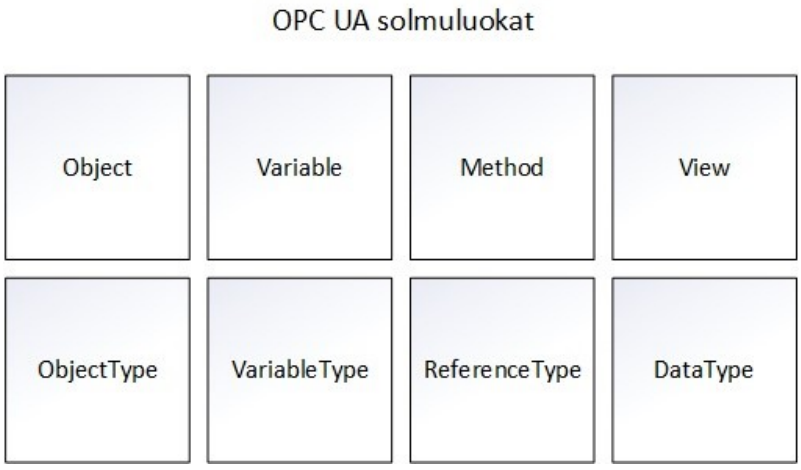
OPC UA:n keskeisimpiä tietomallinnukseen liittyviä konsepteja ovat solmut (engl. node) ja niiden väliset viittaukset (engl. reference). Solmuja ovat kaikki olioperustaiset yksiköt, kuten muuttujat, tietotyypit tai oliot. Kuva 7 esittää OPC UA:n solmua. Jokaisella solmulla on attribuuttina yksilöllinen solmutunniste (engl. NodeId). Viittauksilla pystytään luomaan

solmujen välille assosiaatioita ja asiayhteyksiä. Tiedosta voidaan luoda hyvin verkottunutta, koska solmut voidaan liittää toisiinsa usein eri tavoin. Samalla mahdollistetaan asioiden erilaiset esitystavat ja semantiikat. [37]



Kuva 7. *OPC UA:n solmut. Muokattu lähteestä [43].*

Solmut kuuluvat johonkin kuvan 8 mukaiseen solmuluokkaan riippuen solmun käyttötarkoituksesta [39]. Solmuluokan mukaan solmulla on tietyt luokkaan kuuluvat attribuutit. OPC UA:n määrittely kuvaa kahdeksan solmuluokkaa, jotka perustuvat pohjasolmuluokkaan (engl. BaseNodeclass). Tuolloin kaikilla solmuluokilla ja solmuilla ovat samat perusattribuutit. Attribuutit ovat tietoelementtejä, jotka kuvaavat solmut. Kaikki solmujen tiedot, lukuun ottamatta viittausinstanssia, ovat sisällytetty attribuuttien arvoiksi. [43]



Kuva 8. *OPC UA:n solmuluokat. Muokattu lähteestä [44].*

Palvelut

OPC UA on palvelukeskeinen arkkitehtuuriltaan. OPC UA:n kommunikointi tapahtuu asiakkaiden ja palvelimien välillä käyttäen pyyntö-vastaus-mallia, jossa asiakkaat pyytävät palveluita palvelimilta. Palvelut ovat menetelmiä, joilla asiakas pääsee käsiksi OPC UA -palvelimen tietoihin. [37]

OPC UA:n määritelmän kuvaukset palveluista ovat abstrakteja, eli ne eivät määrittele palveluiden yksityiskohtaista toteutustapaa. Määrittelemällä palvelut abstraktisti, mahdollistetaan eri tiedonsiirtomekanismit ja protokollat. [37] OPC UA:n palvelut kuvataan määrittelyn neljännessä osassa [45].

OPC UA:n palvelut ovat jaettu palveluiden käyttötapauksen mukaan eri kategorioihin, joita kutsutaan palvelujoukoiksi (engl. service set). Palvelujoukot määritellään standardin osassa 4. [45] Palvelujoukot ja niihin kuuluvat palvelut esitetään kuvassa 9.

Discovery Service Set • FindServers • FindServersOnNetwork • GetEndpoints • RegisterServer • RegisterServer2	Method Service Set • Call
SecureChannel Service Set • OpenSecureChannel • CloseSecureChannel	Attribute Service Set • Read • HistoryRead • Write • HistoryUpdate
NodeManagement Service Set • AddNodes • AddReferences • DeleteNodes • DeleteReferences	MonitoredItem Service Set • CreateMonitoredItems • ModifyMonitoredItems • SetMonitoringMode • SetTriggering • DeleteMonitoredItems
View Service Set • Browse • BrowseNext • TranslateBrowsePathsToNodeIds	Subscribtion Service Set • CreateSubscription • ModifySubscription • SetPublishingMode • Publish • Republish • TransferSubscriptions • DeleteSubscriptions
Query Service Set • QueryFirst • QueryNext	

Kuva 9. *OPC UA:n palvelujoukot ja niiden palvelut. Muokattu lähteestä [45].*

Palvelujoukkoja on 9. Discovery-palvelujoukkoon kuuluvat palvelut, joilla voidaan hakea palvelimet, SecureChannel-palvelujoukon palveluilla suoritetaan turvakanavan hallinta. NodeManagement-palvelujoukolla voidaan hallinnoida osoiteavaruuden solmuja. Osoiteavaruuteen voidaan lisätä tai poistaa solmuja ja viittauksia. View-palvelujoukon palveluilla voidaan suorittaa hakuja. Method-palvelujoukon palvelulla *Call* voidaan kutsua palvelimen metodia. Attribute-palvelujoukolla voidaan käsitellä attribuutteja, kuten kirjoittaa tai lukea attribuuttien arvoja. MonitoredItem-palvelujoukon palveluilla voidaan määritellä valvottavia tapahtumia. Subscription-palvelujoukon palveluilla voidaan hallinnoida ja määritellä asiakkaan tekemiä tilauksia. [45]

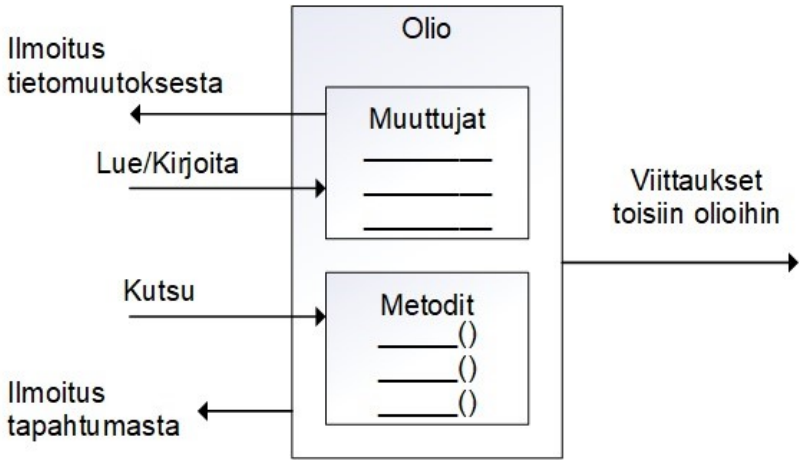
Tietotyypit

OPC UA:n määrittelyiden osa 6 kuvaa kolme tiedon koodaustapaa: OPC UA Binary, OPC UA XML (engl. Extensible Markup Language) ja OPC UA JSON (engl. JavaScript Object Notation). Määrittelyssä kuvataan, kuinka sanomat rakennetaan jokaisella kuvausmenetelmällä. [46]

OPC UA:n määrittelyjen tiedon koodaustavat käyttävät sisäänrakennettuja tietotyyppejä. Sisäänrakennettuja tietotyyppejä käytetään rakenteiden, taulukoiden ja viestien luonnissa [46]. Tietotyypit ovat lueteltu liitteen A taulukossa 1.

Osoiteavaruus

OPC UA:n osoiteavaruuden yksi tärkeimmistä tavoitteista on olioiden esittäminen palvelimella asiakkaille standardoidusti [43,44]. Tiedon yhtenäiseen esittämiseen käytetään oliomallia (engl. object model). Oliomalli määrittelee oliot muuttujien ja metodien avulla. [43] Oliomallin rakenne ilmenee kuvasta 10.

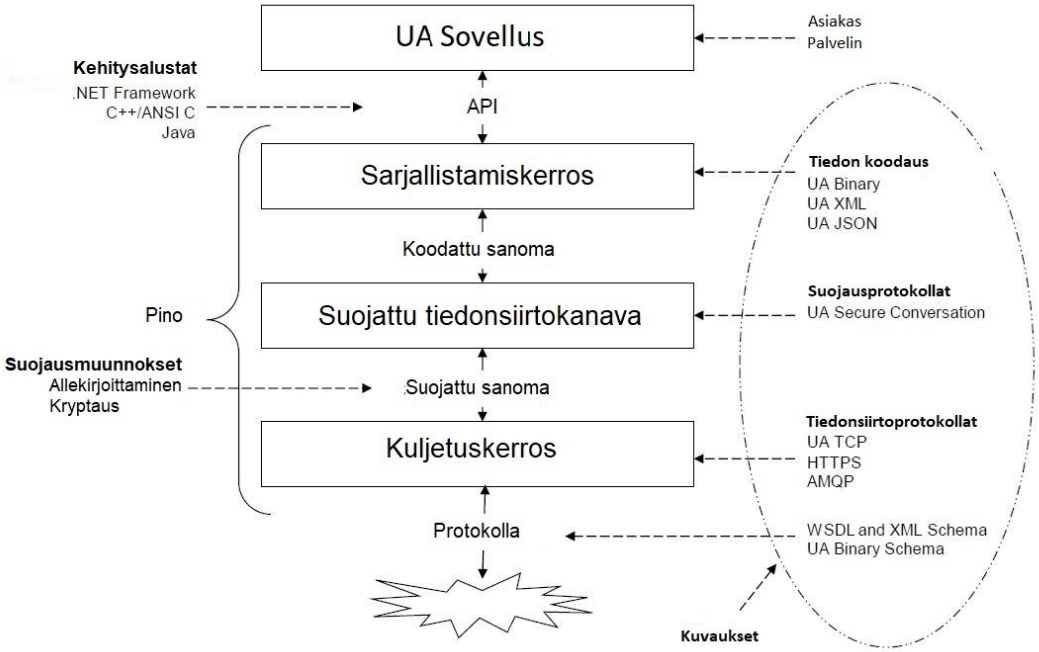


Kuva 10. *OPC UA oliomalli. Muokattu lähteestä [43].*

Oliomallin eri elementit esitetään osoiteavaruudessa solmuina. Solmut kuvataan attribuuttien avulla ja eri solmut yhdistetään toisiinsa viittauksin kuvan 7 mukaisesti. Tätä rakennetta kutsutaan solmumalliksi. Jokainen osoiteavaruuden solmu on solmuluokan instanssi. [43,44]

Kommunikointipino

OPC UA:n kommunikointi koostuu kolmesta pääkerroksesta: sarjallistaminen (engl. serialization), suojattu tiedonsiirtokanava (engl. secure channel), siirtokerros (engl. transport layer). OPC UA:n pino esitetään kuvassa 11, jossa esitetään myös kommunikoinnin kerrokset. [46]



Kuva 11. *OPC UA:n pino ilmaisee OPC UA:n kommunikoinnin rakentumisen. Muokattu lähteestä [46].*

Sarjallistamiskerroksessa suoritetaan sanomien sarjallistaminen sekä sen purku. Kun sovellus lähettää sanoman käyttäen ohjelmointirajapintaa, API (engl. Application programmin interface), sanoma sarjallistetaan OPC UA:n spesifikaatioiden määrittämällä tavalla. Sarjallistettu sanoma lähetetään eteenpäin suojattuun tiedonsiirtokanavaan. [37]

Sanomat, jotka tulevat suojattuun tiedonsiirtokanavaan sarjallistamiskerrokselta, suojataan. Riippuen konfiguraatiosta lähetettävät sanomat joko allekirjoitetaan, salataan tai tehdään molemmat. [46] Sanoman suojaus on mahdollista ottaa pois käytöstä, jos järjestelmää käytetään eristetyssä ympäristössä ja suojausta ei vaadita [37].

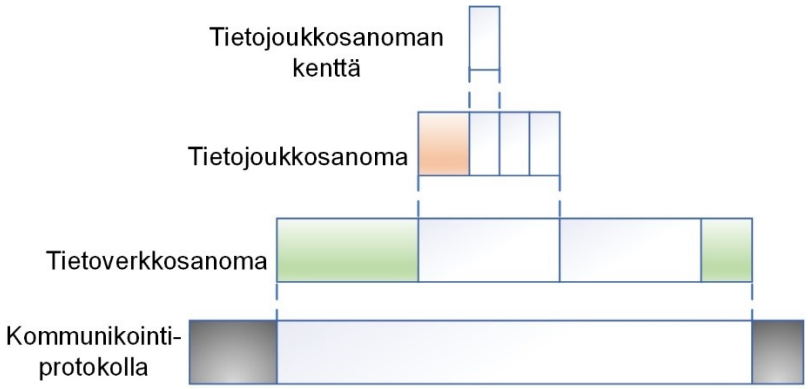
Siirtokerros välittää ja vastaanottaa sanomat sekä suorittaa virheidenkäsittelyn. Kun sanoma lähetetään, siihen lisätään otsikkokentät. Otsikkokentät sisältävät tiedon muun muassa sanomat tyypistä tai pituudesta. Kun sanoma vastaanotetaan siirtokerrokseen, se vahvistaa sanoman muotoilun, tarkistaa voidaanko sanoma tunnistaa tai onko se liian pitkä, ennen kuin se välitetään suojatulle siirtokanavalle. [37]

OPC UA PubSub

OPC UA määrittelyn uusimman osan 14, PubSub, tarkoitus on irti kytkeä OPC UA -sovellukset määrittelemällä sovelluksille julkaisijoiden ja tilaajien roolit. OPC UA PubSub -määrittelyn mukaan sanomat välitetään julkaisijan ja yhden tai useamman tilaajan välillä. Tilaajien määrä ei vaikuta julkaisijoihin, jolloin määrittelystä saadaan hyvin soveltuva järjestelmiin, joissa vaaditaan skaalautuvuutta tai riippumattomuutta sijainnista. [47]

PubSubin käyttökohteiksi on ajateltu olevan muun muassa ohjainten tai HMI:n ja ohjaimien välinen kommunikointi, asynkroniset tehtävät, toimilaitteiden tai anturitiedon tallentaminen HMI-järjestelmään tai tietokantaan. Lisäksi PubSubia voi käyttää palveluita tai laitteita edustavat OPC UA -palvelimet, jotka lähettävät tietoja pilvisovelluksiin. [47]

PubSub-määrittelyä jatkokehitetään aktiivisesti. OPC UA:han on kehitteillä määrittely, jossa TSN-tekniikoita hyödynnetään PubSub-määrittelyn kanssa [48]. Tällä hetkellä kuitenkin on jo olemassa avoimen lähdekoodin OPC UA PubSub käyttäen TSN-tekniikoita. TSN-tekniikkaa käyttämällä mahdollistetaan deterministinen tiedonvälitys OPC UA -sovellusten välillä. [10] Lisäksi yhtenä OPC UA:n ja PubSub-pohjaisen kommunikoinnin tulevaisuuden näkymänä pidetään determinististä PubSub-kommunikointia 5G-verkossa [48].



Kuva 12. *OPC UA PubSub -sanoman kerrokset. Muokattu lähteestä [47].*

Kuva 12 esittää PubSub sanoman monikerroksista rakennetta. Julkaisut koostuvat useasta kerroksesta, joista alimpana kerroksena on käytetty kommunikointiprotokolla, kuten MQTT, AMQP tai OPC UA UDP. Sanoma pitää sisällään yhden tietoverkkosanoman (engl. Network Message), joka sisältää sanoman metatietoa (kuvassa 12 tietoverkkosanoman alku), sekä yhden tai useamman tietojoukkosanoman (engl. DataSetMessage). Tietojoukkosanoman kehys sisältää tiedon julkaisijasta sekä turvalliseen kommunikointiin liittyvää tietoa. Lisäksi tietojoukkosanomat sisältävät kenttiä, joissa sijaitsee sanoman varsinainen tietosisältö, kuten mittaustietojen arvot, tilat ja aikaleimat. Tällä tavoin yhdessä julkaisussa voidaan kuljettaa useita erilaisia sekä eri tyyppisiä tietoja tilaajille. [47]

PubSub-määrittely kuvaa tiedonsiirron käyttämällä AMQP-, MQTT- tai OPC UA UDP -kommunikointiprotokollia. Näiden käyttöä varten määrittely kuvaa, kuinka sanomat muodostetaan, tai mitä koodauksia voidaan käyttää. AMQP- ja MQTT-protokollien tapauksissa sanomat voidaan muodostaa esimerkiksi JSON-muotoa (engl. JavaScript Object

Notation) käyttäen tai OPC UA Binary-koodauksella. OPC UA UDP-protokollaa käytetään UADP-sanomien (engl. UA Datagram Protocol) siirtämiseen. UADP-protokolla käyttää optimoitua UA Binary koodausta. [47]

Tietoturva PubSubissa toteutetaan erillisellä turva-avainpalvelulla (engl. Security Key Service). Turva-avainpalvelu tuottaa avaimet sanomien tietoturvan luomiseksi. Julkaisijat käyttävät palvelua sanomien allekirjoittamiseen ja salaamiseen. Tilaajat käyttävät palvelua allekirjoitusten varmistamiseen sekä salauksen purkamiseen. Palvelu voi olla saatavilla tietoverkossa tai palvelu voi olla toteutettuna julkaisijaan. Turva-avainpalvelua voidaan käyttää myös AMPQ- ja MQTT-protokollilla silloin, kun tiedon siirtämiseen käytetään UADP-tietoverkkosanomia. Jos tiedonsiirtoon käytetään JSON-muotoa, tietoturva toteutetaan esimerkiksi TLS-salauksella (engl. Transport Layer Security) ja välittäjien tietoturvamekanismeilla. [47]

4. IOT-KOMMUNIKOINTIPROTOKOLLAT JA STANDARDIT

Tässä luvussa käydään läpi tutkimuksen kohteena olevia IoT-protokollia ja standardeja. Protokollista käydään läpi niiden heikkouksia, vahvuuksia sekä ominaisuuksia, jotka vaikuttavat protokollan valintaan.

4.1 AMQP

AMQP (engl. The Advanced Message Queuing Protocol) on avoin tiedonsiirron standardi, joka on kehitetty pankkiliiketoiminnan sovellusten viestien välittämiseen [49]. Protokolla on kehitetty pitäen mielessä sanomien toimituksen luotettavuus, nopea toimitus suurelle määrälle asiakkaita ja tiedostojen turvallinen siirto mahdollistaen yksinkertaisen palomuurien konfiguroinnin [50]. AMQP:n standardin uusin versio on 1.0 ja se on OASIS-järjestön (engl. Organization for the Advancement of Structured Information Standards) hyväksymä ja ylläpitämä. Standardi on hyväksytty myös ISO-standardiksi ISO/IEC 19464:2014 (engl. International Organization for Standardization/International Electrotechnical Commission). [51]

Versio 1.0 on kokenut suuria muutoksia vanhemmasta versiosta AMQP 0.9.1:stä. Muutoksia koki muun muassa protokollan topologiamalli. Lisäksi standardiin lisättiin tietotyypit. [52,53] Muutosten takia useat lähteet pitävät AMPQ:n 0.9.1 ja 1.0 versioita eri protokollina. [19,54] Mittavien muutoksien takia protokollan käyttäjän on tiedostettava käytettävän AMQP-version tuki. Tämä tutkimus keskittyy AMQP:n uudempaan versioon 1.0.

AMQP-protokollassa IoT-järjestelmien viestintään useita tärkeitä ominaisuuksia. Näihin lukeutuu muun muassa tietoturvan tuki, vuonohjaus, tiedon koodaus, kommunikointi käyttäen välittäjää tai suoraan päätepisteiden välillä. AMQP mahdollistaa useiden eri kommunikointimallien käytön. AMQP tukee kommunikointiosapuolten irtikytkennän sanomajonoilla tai julkaise-tilaa-mallilla. Myös säiliöiden, eli AMQP-protokollaa käyttävien sovellusten, väliset transaktiot ovat tuettuja. [19]

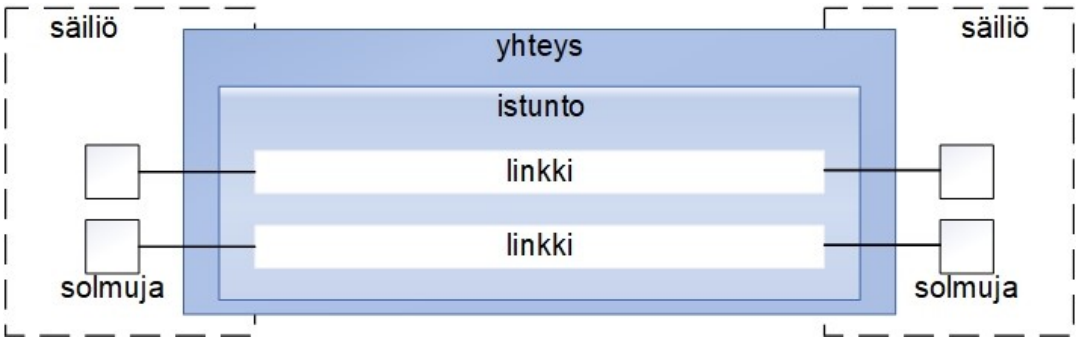
AMQP-protokolla koostuu kolmesta pääkerroksesta. Alin taso määrittelee protokollan, jolla välitetään sanomia verkossa kahden pisteen välillä, toisessa osassa määritellään sanoman muoto sekä sen koodaus ja viimeisessä määritellään välittäjän palveluiden semantiikka. Standardi käsittelee seuraavat protokollan pääosat: koodaus, tiedonsiirto, sanomat, transaktiot ja tietoturva. [52]

AMQP-protokolla on joustava ja se on saanut jalansijaa IoT-protokollana. Sen määrittely ja rakenneosat ovat sopivia IoT-viestinnän käyttötapauksiin. AMQP:ssa on lukuisia ominaisuuksia, mutta toisaalta tästä seuraa kompleksisuutta. Tämän takia tuen toteuttaminen ominaisuuksiltaan rajoittuneisiin laitteisiin voi olla hankalaa. [19]

Arkkitehtuuri

AMQP:n version 0.9.1 nojautuu vielä vahvasti julkaise-tilaa-arkkitehtuuriin, jossa välittäjä saa julkaisijoilta sanomat. Julkaisijat lähettävät sanomat vaihteeseen (engl. exchange), joka jakaa sanomat aiheen mukaisiin sanomajonoihin. Sanomajonoista välittäjä välittää tilaajille tilatut sanomat. [53] Version 1.0 AMQP-protokollalla voidaan toteuttaa tavanomaisia tuottaja-kuluttaja-suunnittelumallin mukaisia arkkitehtuureja, joiden sanoman välityksen tekee välittäjä. Kommunikointi on mahdollista toteuttaa myös suoraan kahden asiakassovelluksen välillä ilman välittäjiä. Tiedonvaihtoa ei ole sidottu uudessa standardissa enää tiettyyn malliin tai topologiaan. [19]

AMQP:n version 1.0 kommunikoinnin arkkitehtuuri koostuu säiliöistä. Säiliöt voivat toimia asiakkaina tai välittäjinä. Jokainen asiakas tai välittäjä voivat sisältää solmuja. Solmut ovat joko tuottajia, kuluttajia tai jonoja. AMQP:n sanomien välitys tapahtuu solmujen välillä. Kahden solmun välille muodostetaan yksisuuntainen linkki. Linkit toimivat joko tiedon vastaanottajina tai lähettäjinä. Linkin pääte on yhteydessä istunnon päätteeseen. [52] Kuvassa 13 esitetään AMQP-protokollan rakenneosat: säiliöt, jotka sisältävät useita solmuja, yhteydet säiliöiden välillä sisältäen yhden tai useamman istunnon ja linkit solmujen välillä. [19]



Kuva 13. *AMQP kommunikointi koostuu useasta rakenneosasta. Muokattu lähteestä [19].*

Sanomat

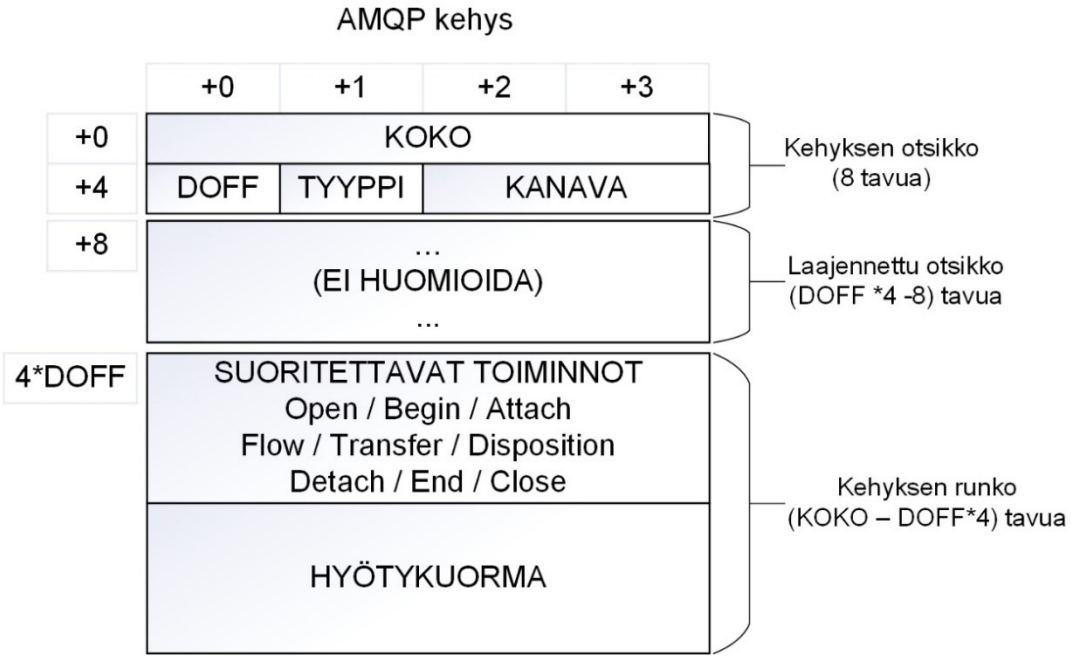
AMQP-protokollassa käytetään 9 erilaista kehysrunkoa. Kullakin rungolla on oma käyttötarkoitus, kuten avata yhteys tai sanomien vaihdon aloitus. Kehysrungon tyypin mu-

kaan kehyksen käsittely tai sisältö tarkastetaan joko yhteystasolla, istunnossa tai linkissä. Kehysrungot, niiden käsittely, tarkastaja ja kuvaus esitetään liitteen B taulukossa 1. [52]

Liitteen B taulukon 1 mukaisesti yhteystaso avaa tai sulkee yhteyden ja tarkastaa istunnon aloituksen sekä lopetuksen. Istuntokerros käsittelee istunnon aloituksen ja lopetuksen sekä tarkastaa sanomien siirtoon liittyvät kehykset. Linkki käsittelee sanomien siirtoon liittyvät kehykset. [52]

Sanomarakenne

AMQP:n sanomarakenne esitetään kuvassa 14. AMQP-protokollan kehykset koostuvat kolmesta osasta: kehyksen otsikosta, laajennetusta otsikosta ja kehyksen rungosta. [52]



Kuva 14. *AMQP-sanoman rakenne salaamattomassa sanomassa. Muokattu lähteestä [52].*

Kehyksen otsikko on 8 tavun mittainen rakenne ja se esiintyy jokaisen kehyksen alussa. Otsikko sisältää pakolliset tiedot, joiden avulla voidaan koota loput koko kehyksestä. Otsikon ensimmäinen kenttä kertoo kehyksen koon ilmaistuna 32-bittisenä kokonaislukuna. Tavun mittainen DOFF-kenttä (engl. data offset) ilmaisee, mistä sanoman kohdasta alkaa kehyksen runko. Tyyppi kertoo sanoman muotoilun ja merkityksen. Tyypillä ilmaistaan esimerkiksi, että sanoma on AMQP-sanoma tai SASL-salattu (engl. Simple Authentication and Security Layer). [52]

Laajennettu otsikko on kenttä, jonka pituus vaihtelee ja jonka käyttö sekä sisältö riippuvat kehyksen tyypistä. Laajennettu otsikko on varattu tulevaisuuden tarpeita varten ja nykyi-

sin version 1.0 spesifikaatiossa AMQP-sanomassa kentällä ei ole käyttöä. Myös kehyksen rungon pituus vaihtelee. Rungon alussa määritellään sanoman tehtävä, eli kehysrungon tyyppi, jonka jälkeen seuraa sanoman hyötykuorma. Kehysrungon tyypin määrittelyyn sisältyy sanoman määrittelyjä, joilla ilmaistaan sanoman eri ominaisuuksia. [52]

Tietoturva

AMQP:n määrittelyissä esitetään tiedonsiirron suojaustavoiksi TLS-suojausta tai SASL-mekanismien käyttöä. AMQP:n standardi määrittelee, kuinka toteuttaa sanomakehykset siten, että SASL-suojausmekanismien käyttö onnistuu. [52] TLS-suojaus on määritelty erikseen omassa määrittelyssä [55].

TLS-suojauksien käyttö mahdollistaa tietoliikenteen salauksen sekä kommunikoivien osapuolten autentikoinnin. Parien autentikointi tapahtuu käyttämällä X.509-varmenteita. Ennen kuin TLS-kommunikointi aloitetaan, jokainen osapuoli lähettää protokollan otsikon. Otsikko koostuu protokollan nimestä ”AMQP”, tunnistenumerosta ja TLS-versionumeroista ja revisionumerosta. [52]

Tietyissä tapauksissa palomuuria käytettäessä, ei ole mahdollista suorittaa TLS-suojausta edellä mainitulla tavalla. Riippuen palomuurin suojauksen tasosta, palomuuri saattaa tulkita paketin ei-TLS-protokollan mukaiseksi, koska protokollasta lähetetään otsikkosanoma ennen suojausta. Tällöin TLS-suojaukseen voidaan käyttää esimerkiksi TLS-palvelimia, jolloin otsikkotietoja ei tarvitse lähettää. [52]

SASL-menetelmiä käyttämällä AMQP-sanoman muodostaminen ja kommunikointi muuttuu. Jotta SASL-kerroksen muodostaminen onnistuu, solmun täytyy lähettää ensin protokollan otsikko samoin kuin TLS-salauksessa, mutta SASL-mekanismien versionumeroilla. Tämän jälkeen suoritetaan SASL-salauksien kättelyt. [52]

SASL-menetelmät määritellään SASL-palvelimelle. Palvelin ilmoittaa asiakaslaitteille tuetut autentikointimenetelmät erillisellä sanomalla. Solmu, joka kommunikoi palvelimen kanssa, valitsee yhden tuetun menetelmän ja aloittaa SASL-kättelyt. [49] SASL-suojausmekanismit määritellään omassa määrittelyssään [56].

Luotettavuus

AMQP:ssä sanomien lähetystä ja niiden vastaanottoa valvotaan sanomaan sisällytetyillä tiedoilla sekä toimitettu-lipulla (engl. settled). Toimitusta pidetään toimittamattomana lähetyksen jälkeen siihen asti, kunnes lähettäjä tai vastaanottaja merkkaa sanoman toimitetuksi. Tähän mekanismiin perustuen AMQP:ssä toteutetaan sanoman toimittamiseen erilaisia luotettavuuden tasoja. Sanomien vaihdon luotettavuuden tasot AMQP:ssä ovat: Vain kerran (engl. At-Most-Once), Ainakin kerran (engl. At-Least-Once) ja Täsmälleen

kerran (Exactly-Once). Näiden toteutus perustuu toimitettu-lipun käsittelyyn sanomien vaihdon eri vaiheissa. [52]

Vain kerran -luotettavuuden taso toteutetaan toimitettu-lipun käsittelyllä siten, että sovellus asettaa toimitettu-tilatiedon ennen lähetystä. Tuolloin lähettäjä ei talleta mitään lähetyksestä eikä yritä uudelleenlähetystä. Vastaanottaja ei vastaa sanomaan, koska sanoma on toimitettu eli käsitelty. [52]

Ainakin kerran -luotettavuuden taso toteutetaan siten, että vastaanottaja määrittelee sanoman toimitetuksi heti sen vastaanotettuaan. Tällöin vastaanottaja lähettää vastauksen alkuperäiselle lähettäjälle. Jos vastaussanoma katoaa, lähettäjä yrittää toimittaa alkuperäisen sanoman uudelleen, koska ei ole tiedossa, että sanoman tila on toimitettu. [52]

Täsmälleen kerran -luotettavuuden taso saavutetaan siten, että lähettäjä lähettää sanoman, jonka toimitettu-lippua ei ole asetettu. Vastaanottaja vastaa toimitukseen sanomalla, jossa ilmaistaan, että alkuperäinen tieto on vastaanotettu. Tämän jälkeen alkuperäinen lähettäjä asettaa toimitettu-tilatiedon, poistaa muistista sanoman tunnisteen ja lähettää sanoman, jossa on alkuperäisen sanoman tunnisteella asetettu toimitettu-tilatieto. [52]

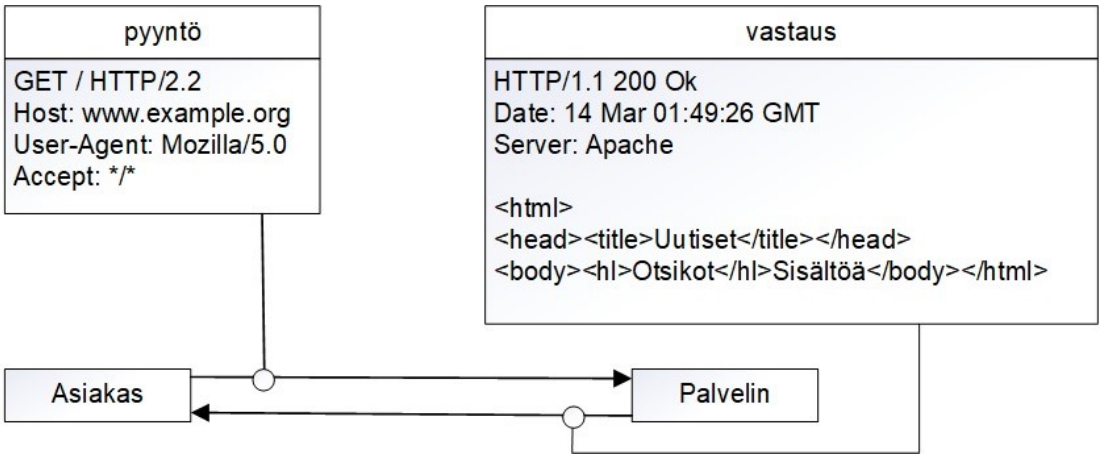
4.2 HTTP

HTTP on sovellustason protokolla, jota käytetään pääsääntöisesti web-sovellusten kommunikointiin. HTTP-protokolla on ollut käytössä jo vuodesta 1990, mutta yleisesti käytetty HTTP/1.1 määrittely julkaistiin vuonna 1997. Tähän määrittelyyn tuli päivitys vuonna 1999. HTTP-protokolla kehittää IETF (engl. Internet Engineering Task Force). Tällä hetkellä viimeisin HTTP:n standardi, HTTP/2, on vuodelta 2015. HTTP on hyvin laajasti tuettu ja siksi useasti käytössä web-sovelluksissa. [19,57] HTTP:n suosio ja laaja käyttö tekee siitä suositun myös IoT-käytössä. [19]

HTTP:n etu on protokollan joustavuus. Pyyntö-vastaussanomiin voidaan lisätä otsikkotietoja pyyntöihin ja vastauksiin. Protokollassa on useita standardoituja otsikoita, jotka ovat yleisesti hyväksyttyjä, kuten autentikointi, välimuistin hallinta ja asiakkaiden mieltymykset koodauksessa. HTTP:n otsikot ovat laajassa käytössä web-selaimissa ja -palvelimissa. [19]

Arkkitehtuuri

HTTP käyttää yhteydellistä asiakas-palvelin-arkkitehtuuria. HTTP:ssä palvelimet vastaavat asiakkaan muodostamiin kyselyihin. Palvelin voi käyttää välityspalvelimia tiedon välittämiseen asiakkaalle. [57] Kuva 15 esittää HTTP:ssä käytettävää pyyntö-vastaus-mallia. Asiakas lähettää pyynnön palvelimelle ja palvelin vastaa käsiteltyään pyynnön. [19]



Kuva 15. *HTTP:n version 1.1 esimerkki pyyntö-vastausmallista. Muokattu lähteestä [19].*

Sanomarakenne

HTTP:n version 1.1 ja version 2 sanomarakenteessa on eroja. Version 1.1 otsikon sisältö ja pituus riippuvat sanomatyypistä sekä sanomat ovat tekstipohjaisia. Version 2 otsikkokentän muodostuminen määritellään yksityiskohtaisesti. Lisäksi version 2 protokolla on binääriprotokolla, eli siirrettävän sanoman muoto ei ole selkokielistä. [57]

Version 1.1 pyyntösanomat koostuvat seuraavista pääosista: Pyyntörivi, yleinen otsikko, pyyntöotsikko, entiteettiotsikko ja sanomarunko. Pyyntörivi sisältää tiedon resurssille käytettävästä metodista (kuvassa 15 pyynnön GET), metodin identifioivan URI:n (engl. Uniform Resource Identifier) sekä käytettävän HTTP-protokollan versiotiedon. Yleinen otsikko sisältää kommunikoinnille aseteltavia parametreja tai ominaisuuksia, jotka ovat käytössä sekä pyynnöissä että vastauksissa. Pyyntöotsikko sisältää palvelimelle välitettäviä lisätietoja pyynnöstä tai asiakkaasta. Entiteettiotsikko määrittelee entiteettirungon metainformaatiota, kuten koodauksen tai sisällön kielen. Sanomarunko sisältää entiteettirungon. Entiteettirunko sisältää sanoman hyötykuorman. [58]

Version 2 sanomarakenne koostuu 9 tavun mittaisesta otsikkokentästä. Otsikkokentän aluksi kerrotaan sanoman pituus sekä tyyppi. Tämän jälkeen sanomaan sisällytetään lippubittejä, joiden merkitys vaihtelee sanoman tyypin mukaan. Lisäksi otsikkoon kuuluu sanomavirran tunniste. Otsikkokentän jälkeen asetetaan sanoman hyötykuorma. Määrittelyssä rajataan sanomien maksimikooksi vähintään 16 384 tavua tai enintään

16 777 215 tavua. Laite, joka käyttää protokollaa määrittelyn mukaisesti, pitää pystyä käsittelemään 16 384 tavun sanomia. [59]

Jokainen HTTP-pyyntö sisältää HTTP-metodin, johon palvelin vastaa tietyllä toimenpiteellä. Yleisimmät metodit ovat GET ja POST. Jäljellä olevia HTTP metodeja käytetään yleisimmin RESTful (engl. Representational State Transfer) ohjelmointirajapinnoissa. HTTP:n metodit ja niiden kuvaukset esitetään liitteen B taulukossa 2. [19]

Vastaussanomat koostuvat tilarivistä, yleisestä otsikosta, vastausotsikosta, entiteettiotsikosta ja sanoman rungosta. Tilarivi sisältää tiedon protokollan versiosta sekä vastauksen tilan. Tilatiedot ovat yleisesti hyväksyttyjä. Taulukossa 1 esitetään tilatietojen kategorointi. Yleinen otsikko on samanlainen pyynnöissä ja vastauksissa. Vastausotsikko sisältää asiakkaalle välitettäviä lisätietoja. Entiteettiotsikko määrittelee pyynnön tavoin entiteettirungon metainformaatiota. [58]

Taulukko 1. *HTTP-vastaustilatietojen kategoriat. Muokattu lähteestä [19].*

Muoto	**Kategoria**	**Tarkoitus**
1xx	Tiedottava	Yleinen tieto, joka ei kerro virhetilanteista tai tapahtuman onnistumisesta.
2xx	Onnistuminen	Pyyntö onnistui ja vastauskoodi tuottaa lisätietoa onnistumisesta.
3xx	Uudelleenohjaus	Pyydetty resurssi ei ole saatavilla ja uudelleenohjausta tarvitaan.
4xx	Asiakkaan virhe	Asiakas suoritti virheellisen pyynnön ja operaatiota ei voitu suorittaa.
5xx	Palvelimen virhe	Virhe tapahtui palvelimen päässä.

Tietoturva

HTTP:n tietoturva perustuu TLS-standardiin kommunikoinnin salaamisessa asiakkaan ja palvelimen välillä. Tämän lisäksi HTTP-protokollan määrityksessä määritellään kaksi autentikointimekanismia: perusautentikointi ja digest-autentikointi. [19]

Perusautentikoinnissa lähetetään käyttäjänimi ja salasana base64-koodattuna tekstinä palvelimelle. Autentikointi ei salaa salasanaa tai käyttäjänimeä, jolloin tunnukset ovat sanomasta saatavissa. Tästä syystä autentikointi vaatii myös TLS-salauksen käyttämistä. [60]

Digest-autentikoinnissa käytetään tiivistettä. Asiakas saa tiivisteen palvelimelta. Asiakas laskee MD5-algoritmia (engl. message digest) käyttäen käyttäjätunnuksesta, salasanasta ja tiivisteestä lukuarvon, jonka asiakas lähettää palvelimelle. Käyttämällä tiivistettä, palvelin voi jälleenrakentaa oikean käyttäjänimen ja salasanan. Tuolloin käyttäjänimeä ja salasanaa ei lähetetä selkokielisenä. [19,60]

Luotettavuus

HTTP-protokollaan ei ole määritelty juurikaan mekanismeja, esimerkiksi palvelun laatutasoja, luotettavuuden takaamiseksi. HTTP/1.1 protokollassa HTTP asiakas ei pysty toistamaan epäidempotentteja pyyntöjä virheen sattuessa, koska virheen laatua ei pystytä määrittämään. HTTP/2 toteuttaa kaksi mekanismia, joilla voidaan kertoa asiakkaalle, että pyyntöä ei käsitelty. [59]

HTTP/2.0

Päivityksen HTTP/2 suurimmat kehityskohdat tekstipohjaisesta HTTP/1.1:stä ovat binäärinen kommunikointi, pyyntöjen tai vastauksien multipleksaus, otsakkeen pakkaaminen ja muuttuvan resurssin lähetys asiakkaalle. Binäärisessä kommunikoinnissa sanomat ovat jäsennettävissä helpommin ja niistä saadaan kompaktimpia. [19] Sanomat koodataan binääriseen muotoon, mutta sisällön merkitys pysyy samana HTTP/1 ja HTTP/2 protokollissa. Korkean tason sovelluksen ei tarvitse tuntea, kuinka sanoma lähetetään ja siksi ne voivat käsitellä HTTP/1 ja HTTP/2 yhteyksiä samalla tavoin. [61]

HTTP/1 on synkroninen, jossa suoritetaan yksi pyyntö ja vastaus. HTTP/2 sallii useamman pyynnön käsittelyn samanaikaisesti samassa TCP-yhteydessä. Tuolloin käytetään useita tietovirtoja (engl. streams) jokaiselle pyynnölle tai vastaukselle. Tämän mahdollistaa osaltaan protokollan binäärisyys ja kehyksien sisältämät tietovirtojen tunnisteet. [61]

HTTP-otsikkotietoja käytetään pyyntöjen ja vastauksien lisätietojen lähettämisessä. Otsikot voivat olla eri lähetyksissä samanlaiset, jolloin sanomien otsakkeissa on toistoa. HTTP/2 mahdollistaa otsakkeiden pakkaamisen, jossa sanomista poistetaan toistuva tieto. Pakkausalgoritmi mahdollistaa sen, että poistettu tieto voidaan palauttaa pakkauksen purkamisessa, jolloin tietoa ei menetetä. [61]

HTTP/1-protokollassa jokainen resurssi pyydetään erikseen. HTTP/2 mahdollistaa palvelimen lähettää resursseja asiakkaille asiakkaan pyytämättä. [61] Palvelin voi käyttää ominaisuutta tiedon lähettämiseen asiakkaalle resurssien muuttuessa [19].

RESTful ohjelmointirajapinnat

REST määrittelee ohjelmointirajapintojen, APIen, arkkitehtuurityylin, jolla sallitaan asiakkaiden pääsy resursseihin ja niiden muokkaaminen ennalta määritetyillä tilattomilla operaatioilla. Suurin osa RESTful rajapinnoista pohjautuu HTTP-protokollaan ja käyttävät laajasti HTTP-metodeja. RESTful ohjelmointirajapinnoista ei ole standardia, mutta eri koodauksia, kuten JSON tai XML, käytetään HTTP:n, URI:en ja TLS:n kanssa. [19]

4.3 COAP

CoAP-kommunikointiprotokolla on kehitetty IETF:n ja ARM:n yhteistyönä ja se standardoitiin kesäkuussa 2014 avoimeksi standardiksi RFC 7252 (engl. Request for Comments). CoAP-kommunikointiprotokolla on suunniteltu laitteille, jotka ovat rajoittuneita ominaisuuksiltaan, kuten prosessointiteholtaan tai muistiltaan. CoAP käyttää UDP:tä (engl. User Datagram Protocol) tietoliikenneprotokollana. [62] UDP-protokollan käyttö mahdollistaa osaltaan pienen kehyskoon, koska UDP-sanoma ei sisällä synkronointiparametreja, priorisointitietoja tai sekvenssinumeroita. Lisäksi UDP-sanomaa ei tarvitse kuitata, jolloin UDP-protokolla on nopea ja kevyt. [63]

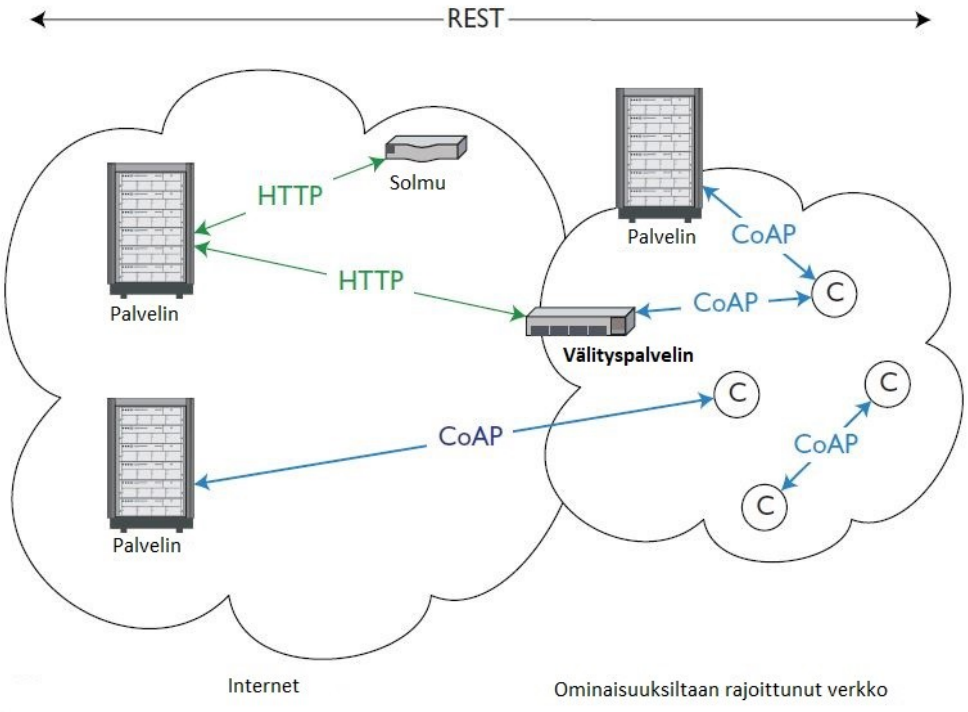
CoAP-protokolla sisältää sisäänrakennetun resurssien paljastamisen, joka sallii asiakkaiden havaita kaikki CoAP-palvelimen tuottamat resurssit. Mekanismi on erityisen hyödyllinen M2M-viestinnässä (engl. Machine-to-Machine). [62] CoAP-asiakas voi muodostaa GET-pyynnön, johon palvelin vastaa. Vastaus sisältää linkkejä saatavilla oleviin resursseihin ja niiden attribuutteihin. Mekanismi toimii myös UDP-monilähetyksellä sallien kaikkien saatavilla olevien resurssien tunnistamisen. [19]

CoAP-protokolla mahdollistaa tarkkailija-mallin (engl. Observer) mukaiset resurssit. RFC 7641 kuvaa CoAP-laajennuksen, joka sallii asiakkaiden tarkkailla resursseja. Kun asiakas lähettää tarkkailupyynnön, kaikki resurssien päivitykset lähetetään palvelimelta asiakkaille. [19] Asiakkaan GET-pyynnössä voidaan osoittaa kiinnostus myöhemmille resurssin päivityksille määrittelemällä tarkkailu sanoman valintoihin. Jos palvelin hyväksyy määrittelyn, asiakkaasta tulee resurssin tarkkailija ja vastaanottaa asynkronisesti sanomat resurssin muuttuessa. Jokainen palvelimen tuottama ilmoitus on rakenteeltaan identtinen verrattuna alkuperäiseen GET-pyynnön vastaukseen. [64]

CoAP-protokollassa on käytettävissä ryhmäkommunikointi ja monilähetys UDP monilähetyksenä IP-monilähetysryhmään. Ominaisuus kuvataan standardissa RFC 7390 CoAP-protokollalle. Ominaisuus ei kuitenkaan ole käytettävissä kaikilla palvelimilla tai, kun käytössä on DTLS-suojaus (engl. Datagram Transport Layer Security). [19]

Arkkitehtuuri

CoAP:n suosituimmat käyttötapaukset ovat anturien tuottamien mittaustietojen tiedonsiirrossa langattomissa ja langallisissa verkoissa. Kuvan 16 mukaisesti CoAP-protokollaa käytetään usein rajoitetuissa sekä paikallisissa verkoissa. Tästä syystä CoAP-protokollaa välitetään käyttäen HTTP-protokollaa tiedonsiirron helpottamiseksi internetissä. [19] CoAP-sanoman välitys on määritelty CoAP-standardissa [62].



Kuva 16. *CoAP-arkkitehtuurissa CoAP-laitteet kommunikoivat keskenään ja välityspalvelimen kautta HTTP-palvelimille. Muokattu lähteestä [64].*

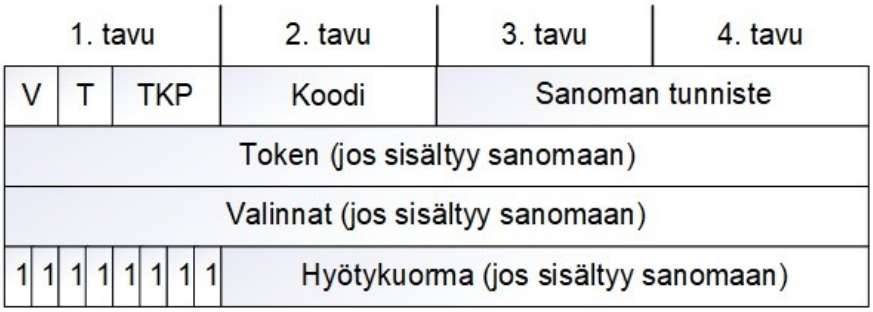
HTTP:ssä ja CoAP:ssa on useita samoja piirteitä, kuten asiakas-palvelin-kommunikointimalli. CoAP on kuitenkin suunniteltu M2M käyttöön, joten laitteet ovat usein sekä asiakkaita että palvelimia. CoAP perustuu REST-arkkitehtuurityyliin. Tuolloin tieto on saatavilla resursseina, jotka tunnistetaan URI:lla. Molemmissa protokollissa, HTTP:ssä ja CoAP:ssa, ovat käytössä samat pyyntömetodit GET, PUT, POST ja DELETE. [19] Vastauskoodit metodeihin ovat samat, mutta CoAP:n vastauskoodin koodaus tapahtuu yhdellä tavulla esitettynä [62]. Vastauskoodien kategoriat lueteltiin aliluvun 4.2 taulukossa 1.

HTTP:n ja CoAP:n erot tulevat ilmi protokollien käyttötarkoituksesta. HTTP:n kommunikointi tapahtuu pyyntö-vastaus-mallisesti, jossa asiakas lähettää pyynnön palvelimelle ja johon palvelin vastaa [58]. Vaikka CoAP-protokolla toimii pyyntö-vastaussemantiikalla, sen on mahdollista toimia myös asynkronisesti tarkkailija-kommunikointimallia käyttäen. [62]

Sanomarakenne

CoAP-protokolla on luotu välittämään pienikokoisia sanomia käyttäen oletusarvoisesti UDP-protokollaa, jossa jokainen CoAP-sanoma varaa yhden UDP-tietosähkeen [62]. Sanomien välitys käyttäen SMS (engl. Short Message Service) [65], TCP [66], SCTP (engl. Stream Control Transmission Protocol) [67] on myös mahdollista.

CoAP-sanomat koodataan yksinkertaiseen binääriseen muotoon [62]. Kuvassa 17 esitetään CoAP-sanoman rakenne.



Kuva 17. *CoAP-sanoman rakenne. Muokattu lähteestä [62].*

Sanoma alkaa neljän tavun mittaisella otsakkeella. Otsikon ensimmäinen tavu sisältää tiedot CoAP-protokollan versiosta (V), sanoman tyypistä (T) ja Token-kentän pituuden (TKP). Tyypillä ilmoitetaan, onko sanoma asetettu vahvistettavaksi, ei-vahvistettavaksi tai onko sanoma kuittaus. Otsikon toisen tavun tiedolla (koodi) ilmaistaan sanoman luonne; onko se pyyntö, vastaus vai virheilmoitus. [62]

Otsikkoa seuraa Token-arvo, jonka pituus voi olla 0–8 tavua. Token-arvolla yhdistetään vastaukset pyyntöihin. Valinnoilla määritellään sanoman ominaisuuksia, kuten kauanko sanomaa voidaan säilyttää välimuistissa. Valintatietoja voi kuulua sanomaan 0–4 tavun verran. Hyötykuorman alkua indikoidaan tavulla 1-arvoisia bittejä. [62]

Tietoturva

CoAP-spesifikaatio määrittelee neljä eri suojauksen tilaa. CoAP:n tietoturva nojautuu spesifikaatiossaan DTLS-suojaukseen, koska tieto välitetään UDP-protokollalla. Suojaustilat ovat: [62]

1. NoSec, jossa ei ole protokollatason suojausta, eli DTLS on pois käytöstä. CoAP:n määrittely ohjeistaa käyttämään muita suojaustekniikoita, jos DTLS on poistettu käytöstä. Tässä tilassa paketit lähetetään käyttäen UDP:tä sekä normaalia CoAP-porttia.
2. Esijaetut avaimet-suojauksen (engl. Pre-shared key) tilassa DTLS on käytössä. On olemassa lista esijaettavista avaimista, jotka sisältävät listan solmuista, joiden

kanssa avaimen haltija voi kommunikoida. Taso voidaan määritellä lisäksi siten, että yhdellä solmulla voi olla vain yksi kommunikointipari.

3. Julkinen avain ilman sertifikaattia-tasossa DTLS on käytössä ja laitteella on epäsymmetrinen avainpari.
4. Sertifikaatti-suojaustilassa DTLS on käytössä ja laitteella on epäsymmetrinen avainpari sekä X.509-sertifikaatti, joka sitoo sen kohteeseen.

DTLS-suojauksella toteutetaan samankaltainen suojaus UDP-tiedonsiirrossa, kuin TLS-suojauksella TCP-tiedonsiirrossa, koska UDP-tiedonsiirrossa ei voida taata pakettien vastaanottamista ja saapumisjärjestystä. Tästä syystä DTLS-suojaukseen on toteutettu mekanismeja, joita suojaaminen vaatii. [68]

Luotettavuus

CoAP-sanomat voivat saapua lähetysjärjestyksestä poikkeavassa järjestyksessä, duplikaatteina tai sanomat voivat kadota ilman, että se huomataan. Näitä asioita varten CoAP-protokollaan on sisällytetty kevyitä luotettavuutta parantavia mekanismeja, ilman että siitä on luotu TCP-liikennöinnin kaltaista. Edellä mainittuja luotettavuutta lisääviä mekanismeja ovat pysähdy-ja-odota uudelleenlähetys sekä duplikaattien tunnistus. [19]

Pysähdy-ja-odota uudelleenlähetyksessä CoAP-sanomat merkataan vahvistettaviksi protokollalipulla. Tuolloin lähettäjän ja vastaanottajan välille otetaan käyttöön hyväksyntämekanismi. Mekanismin käyttö mahdollistaa Ainakin kerran -luotettavuuden tason. Jotta tiedonsiirto ei ruuhkaudu, uudelleenlähetyksien odotusaika kasvaa eksponentiaalisesti. [19,62] Uudelleenlähetystä kontrolloi uudelleenlähetyslaskuri. [62]

Kun käytetään CoAP-protokollaa, sanomien vastaanottaja saattaa saada saman vahvistettavan sanoman useasti, jos esimerkiksi kuittaussanomat ovat jääneet tulematta. Jokaiseen uudelleenlähetykseen asetetaan sama sanomatunniste, joiden avulla duplikaatit tunnistetaan. Tämän tunnisteen perusteella vastaanottaja voi hylätä tulleet duplikaatit. [62]

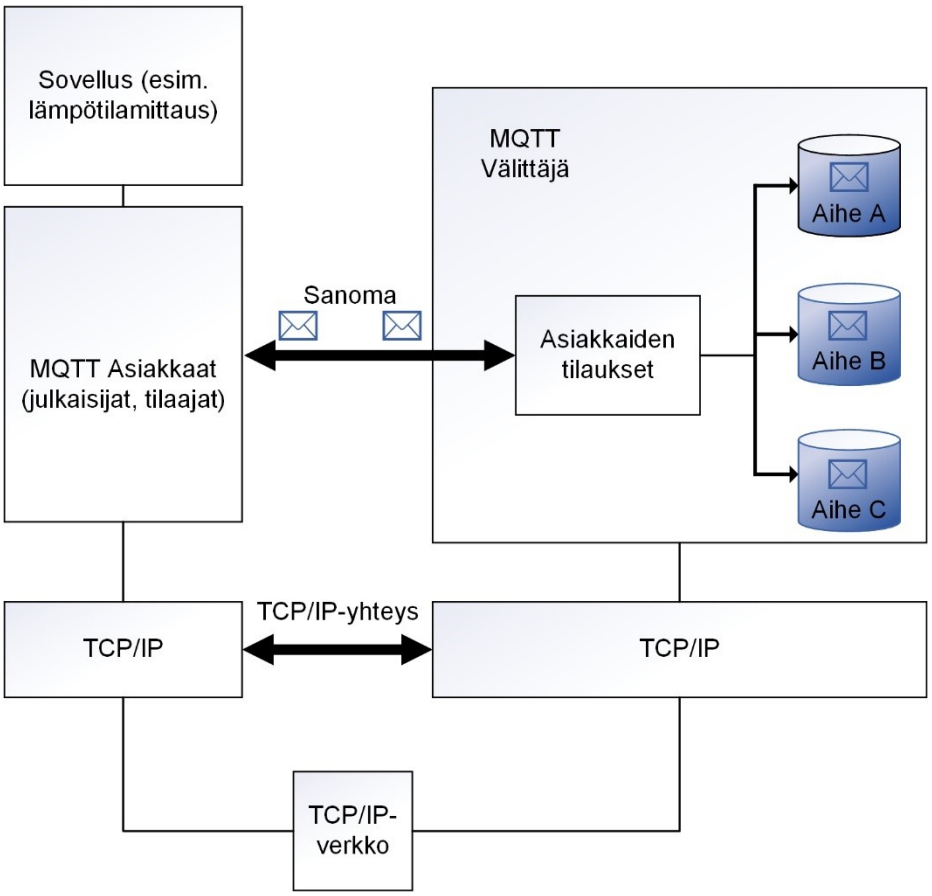
Sanoman voi välittää myös ilman luotettavuutta, jolloin sanomalle ei vaadita kuittaussanomaa. Tätä voidaan käyttää sanomissa, joita toistetaan säännöllisesti, kuten anturien tuottamissa mittaustiedoissa. Näissä tapauksissa sanomat lähetetään ei-vahvistettavina. Tuolloin lähettäjä ei voi tietää, saavuttiko sanoma kohteen. Lähettäjä voi halutessaan lähettää sanomasta usean kopion tai tietoverkko voi luoda sanomasta duplikaatin. Myös näissä tapauksissa sanomaan voidaan asettaa sanomatunniste, jotta vastaanottaja voi tunnistaa duplikaatit. [62]

4.4 MQTT

MQTT (engl. Message Queuing Telemetry Transport) on M2M- sekä IoT-kommunikointiprotokolla. Se on suunniteltu kevyeksi julkaisija-tilaaja-mallin mukaiseen sanomien kuljetukseen. [69] Lisäksi MQTT:tä kuvaillaan helposti käyttöönotettavaksi yksinkertaisen rakenteen takia [70,71]. Yksinkertaisuuden ja mukautuvuuden takia MQTT soveltuu käytettäväksi ominaisuuksiltaan rajoittuneisiin laitteisiin. Lisäksi se on myös suosittu mobiilisovelluksissa [19]. MQTT on TCP/IP-pohjainen. MQTT-standardi on nykyisin OASIS-järjestön ylläpitämä ja kehittämä. Standardista on julkaistu versio 5.0. MQTT-protokollan version 3.1.1 standardi on hyväksytty myös ISO-standardiksi ISO/IEC 20922:2016 [69,72].

Arkkitehtuuri

MQTT käyttää kuvan 18 mukaista arkkitehtuuria, jolla erotetaan sanoman lähettäjä sanoman vastaanottajasta. Sanomavälittäjä toimittaa sanomat tietyille asiakkaille, jotka ovat tilanneet sanomia. MQTT:ssä tilaajien ja julkaisijoiden ei tarvitse olla tietoisia toistensa olemassaolosota, koska kaikki osapuolet kommunikoivat välittäjän kanssa. MQTT:n asiakas voi toimia sekä tilaajana että julkaisijana, jolloin tiedonvaihto voi olla kaksisuuntaista. MQTT-protokolla käyttää sanomien aihepohjaista suodatusta, jonka perusteella tietyt sanomat toimitetaan tietyille asiakkaille. Jokainen sanoma sisältää aiheen (engl. topic), jonka perusteella sanoma välitetään. [19]



Kuva 18. *MQTT:n arkkitehtuuri. Muokattu lähteestä [73].*

Sanomarakenne

MQTT:n sanomapaketteja kutsutaan ohjauspaketeiksi (engl. control packet). Ohjauspaketin rakenne koostuu enimmillään kolmesta kentästä kuvan 19 mukaisesti. Paketti koostuu kiinteästä otsikkokentästä (engl. fixed header) muuttujaotsikkokentästä (engl. variable header) ja hyötykuormasta (engl. payload). *[72]*

Kiinteä otsikkokenttä, sisältyy jokaiseen MQTT ohjauspakettiin
Muuttujaotsikkokenttä, sisältyy joihinkin MQTT ohjauspaketteihin
Hyötykuorma, sisältyy joihinkin MQTT ohjauspaketteihin

Kuva 19. *MQTT ohjauspaketin sanomarakenne. Muokattu lähteestä [72].*

Kiinteä otsikkokenttä on jokaisessa MQTT-sanomassa. Otsikkokenttä koostuu kolmesta osasta. Ensimmäinen tavu sisältää tiedon MQTT-sanoman ohjauspaketin tyypistä sekä lippubittejä (engl. flag bits) joilla määritellään ohjaussanan ominaisuuksia. Ohjauspaketin tyypit on eritelty liitteen B taulukossa 3. Otsikkokentän toisesta tavusta alkaen ilmaistaan paketin jäljellä olevaa kokoa tavuina. Jäljellä oleva pituus ilmaistaan käyttäen 1–4 tavua, joissa tieto esitetään tavun mittaisina kokonaislukuina. [72]

Muuttujaotsikkokenttä ei ole pakollinen otsikkokenttä jokaisessa sanomassa, vaan sen tarve riippuu paketin tyypistä. Otsikkokentän alkuun on varattu kaksi tavua paketin identifiointiin. Identifiointia käytetään, kun MQTT-sanoman palvelunlaadulla, QoS (engl. Quality of Service), on suurempi arvo kuin 0. Tällöin sanoman lähettäjä identifioi paketin. Palvelunlaadusta MQTT-protokollassa kerrotaan tämän aliluvun kohdassa *Luotettavuus*. Muuttujaotsikkokentän lopussa on varaus sanoman ominaisuuksien ilmoittamiseen. [72]

Jotkin ohjauspaketit sisältävät hyötykuorman paketin lopussa. Julkaisu-ohjauspaketeissa (PUBLISH) hyötykuorma sisältää varsinaisen siirrettävän tiedon, eli viestisisällön, jonka lähettäjä haluaa julkaistavan. Hyötykuorman koko voi olla 0 tavua tai suurin paketin koko (268 435 455 tavua), josta vähennetään muuttujaotsikkokenttä. [72]

Edellä käytiin läpi MQTT-sanoman kerrokset ja kenttien mahdolliset koot. Otsikkokenttien yhteenlaskettu minimikoko on 2 tavua ilman muuttujaotsikkokenttää. Julkaisusanomaan sisältyy aina muuttujaotsikkokenttä. Muuttujaotsikkokentän alkuun asetetaan julkaisun aihe, joka esitetään UTF-8 (engl. Unicode Transformation Format) koodattuna merkkijonona. MQTT:n versio 5.0 toi protokollan julkaisuun mahdollisuuden käyttää aiheesta vaihtoehtoista aihetta, aiheen aliasta. Se esitetään kahden tavun mittaisena kokonaislukuna. Aiheen aliasta käyttämällä saadaan pienennettyä julkaisujen sanomakokoa. [72]

Tietoturva

MQTT:n spesifikaatioissa 3.1.1 ja 5.0 määritellään palvelimien ja asiakkaiden yksinkertainen autentikointimenetelmä. Yhteydenluonti (CONNECT) -ohjauspaketti sisältää kentät käyttäjätunnukselle, salasanalle sekä autentikointimetodille. Autentikointimetodilla määritellään millä tavoin autentikointi suoritetaan. Tätä autentikoinnin menettelytapaa määrittelyissä kutsutaan parannelluksi autentikoinniksi (engl. enhanced authentication.) [70,72]

Jos käytetään tavanomaista autentikointia, jossa käytetään käyttäjätunnusta ja salasanaa, suojaus ei ole palvelimen ja asiakkaan kesken symmetrinen. Palvelin voi autentikoida asiakkaan, koska asiakas lähettää luotaessa yhteyden käyttäjätunnuksen ja salasanan, mutta asiakas ei voi autentikoida palvelinta. [70,72] Lisäksi suojaamattomalla yhteydellä salasana, käyttäjätunnus sekä hyötykuorma välitetään selkokielisenä [72,74].

Vaikka MQTT:n standardi ei määrittele tietoturvaominaisuuksia laajalti, määrittelyt sisältävät useita suositeltavia toimenpiteitä tietoturvan lisäämiseksi [72,75]. Lisäksi useissa lähteissä annetaan ohjeistuksia tai työkaluja tietoturvaan liittyen [74-77]. Käytettyjä suojausmekanismeja, jotka mainitaan esimerkiksi standardissa, ovat yhteys salatulla VPN-

yhteydellä (engl. Virtual Private Network) tai luomalla salattu yhteys TLS-salauksella [72].

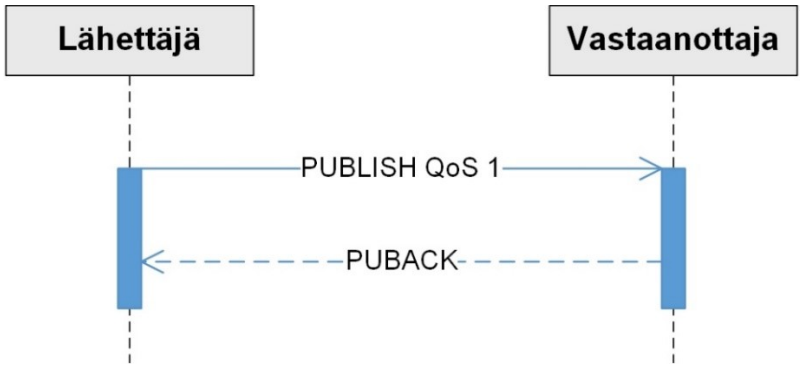
Luotettavuus

MQTT:ssä palvelunlaadulla, QoS, tarkoitetaan tiedon välityksen valvontamekanismeja, joilla määritellään taso, millä varmuudella julkaistu sanoma välitetään eteenpäin. Julkaisun palvelunlaadun tasoa täytyy tarkastella kahdesta näkökulmasta: Julkaisija määrittelee julkaisulle palvelunlaadun tason, kun se lähettää sanoman välittäjälle, mutta tilaaja voi tilauksessa määritellä saman tai pienemmän palvelunlaadun tason. Välittäjä välittää sanoman tilauksen mukaisella tasolla. Asiakkaat valitsevat käytettävän palvelunlaadun tason sen mukaan kuinka luotettava tietoverkko on käytettävissä tai mitä tasoa sovellus edellyttää.[78] Protokollan spesifikaatio määrittelee kolme palvelunlaatutasoa: [72]

0. Vain kerran
1. Ainakin kerran
2. Täsmälleen kerran

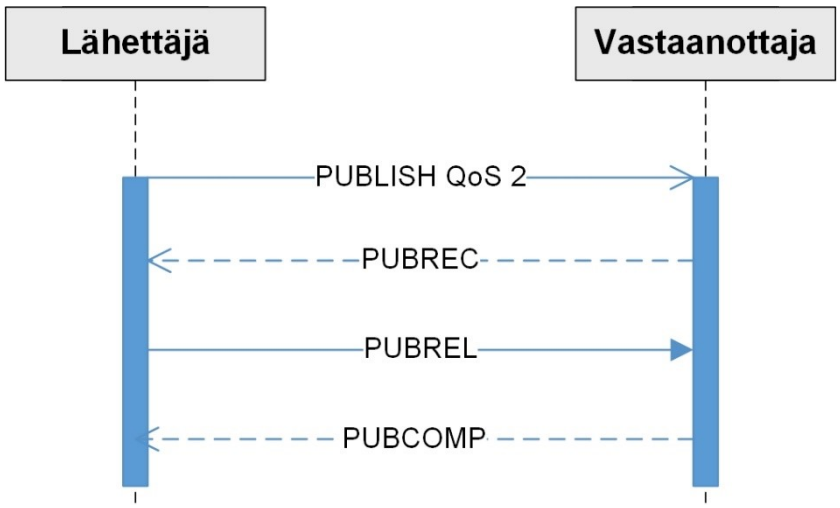
Vain kerran -palvelunlaatutasossa sanoma lähetetään eteenpäin vain kerran, jolloin lähettäjä ei odota vastaanottajalta kuittausta. Sanoma voi saapua vastaanottajalle tai ei ollenkaan. [72] Tällöin sanomalla ei ole varsinaista reaaliaikavaatimusta [78].

Ainakin kerran -palvelunlaatutasossa varmistetaan, että vastaanottaja vastaanottaa sanoman ainakin yhden kerran. Kuvassa 20 esitetään ainakin kerran -palvelunlaatutason sanomienvaihdon toteutus. Kun asiakas julkaisee sanoman, paketti identifioidaan yksilöllisellä pakettitunnisteella. Pakettitunniste sisällytetään muuttujaotsikkokenttään. Vastaanottaja hyväksyy Julkaisu-ohjauspaketin (PUBLISH) vastaanotetuksi julkaisun hyväksyntäohjauspaketilla PUBACK. Jos julkaisija ei saa hyväksyntäsanomaa, julkaisu suoritetaan uudelleen. [72]



Kuva 20. *MQTT julkaisun kättelyt lähettäjän ja vastaanottajan kesken palvelunlaatutasolla ”Ainakin kerran”. Muokattu lähteestä [78].*

Täsmälleen kerran -palvelunlaadussa varmistetaan, että vastaanottaja vastaanottaa julkaisun kerran [72]. Palvelunlaatutaso on turvallisin sanoman perillepääsyn kannalta, mutta myös hitain tapa suorittaa lähetyksiä. Sanoman toimitus varmistetaan kahdella pyyntö/vastaus sanomilla (neliosainen kättely) lähettäjän ja vastaanottajan kesken. Julkaisu tunnistetaan vähintään kerran -palvelunlaadun tavoin pakettitunnisteella. [78] Sanoman julkaisu suoritetaan kuvan 21 mukaisella tavalla. Lähettäjä suorittaa sanoman julkaisun, PUBLISH, palvelunlaatutasolla QoS 2. Vastaanottaja hyväksyy ja ilmoittaa lähettäjälle, että sanoma on vastaanotettu PUBREC-sanomalla. Tämän jälkeen suoritetaan vielä yksi kättelypari, jotta voidaan todeta julkaisun onnistuneen. Sanomiin sisällytetään pakettitunniste, jonka on pysyttävä kättelyiden ajan muuttumattomana. [72]



Kuva 21. *MQTT julkaisun kättelyt lähettäjän ja vastaanottajan kesken palvelunlaatutasolla "Täsmälleen kerran". Muokattu lähteestä [78].*

MQTT-protokolla tarjoaa myös muita sanomien välityksen luotettavuuteen liittyviä ominaisuuksia palvelunlaatutasojen lisäksi. Asiakkaan ja välittäjän istunto voidaan määritellä pysyväksi istunnoksi. Kun asiakas ottaa yhteyden uudelleen välittäjään, aloitetaan istunto välittömästi. Tällöin välittäjä muistaa kaikki istuntojen yksityiskohdat, kuten tilatut aiheet ja sanomat, joita ei ole kuitattu. MQTT-välittäjä asettaa laatutasojen QoS=1 ja QoS=2 sanomat jonoon pysyvien istuntojen asiakkaille silloin, kun asiakas ei ole yhteydessä välittäjään. [19,79]

Sanomat voidaan myös erikseen asettaa pysyväksi. Tuolloin välittäjä säilyttää aiheeseen kuuluvan sanoman. Uusille asiakkaille, jotka tilaavat aiheen, lähetetään heti viimeisin pysyvä sanoma. Asiakkaat saavat heti viimeisimmän sanoman sen sijaan, että odotetaan uuden sanoman saapumista. [19,80]

Epäluotettavat yhteydet luovat tilanteita, joissa toinen TCP-yhteyden osapuoli ei ole saatavissa, ilman että toista osapuolta ilmoitetaan. Tämä voi aiheuttaa asiakkaan tai välittäjän resurssien tuhlausta. MQTT:ssä on sovellustason protokollan yhteydenvalvonta, joka on asiakkaan konfiguroitavissa. Jos kommunikoinnin osapuoli ei lähetä valvonta- tai muuta MQTT-sanomaa määritellyssä aikaikkunassa, välittäjä tai asiakas olettaa, että yhteys on katkennut. [19]

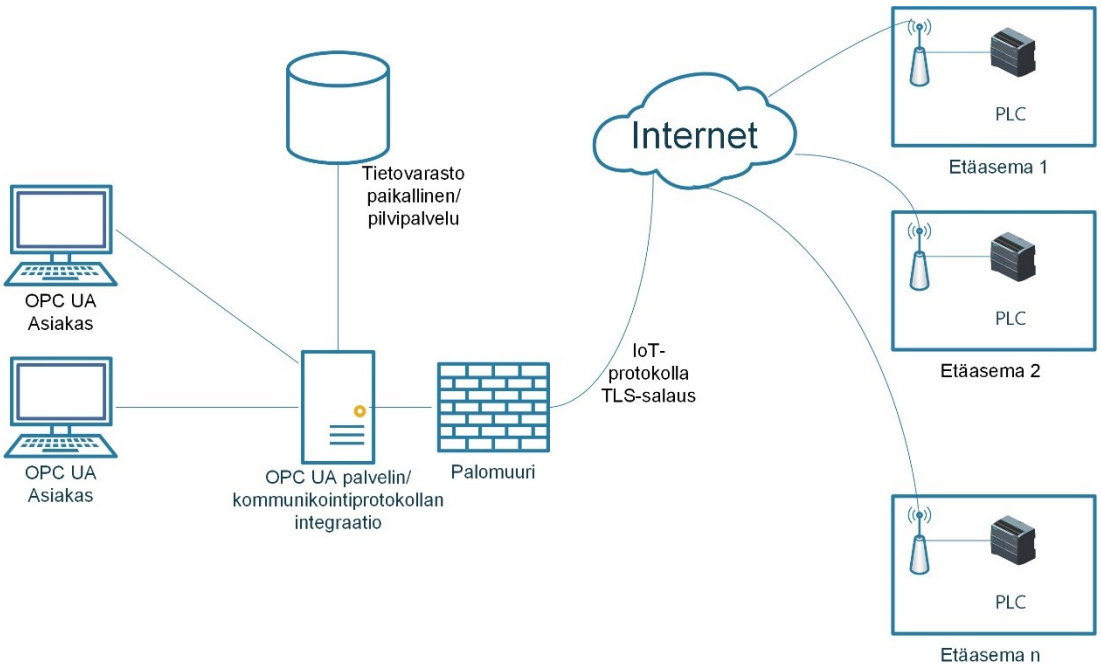
Lisäksi epäluotettavia yhteyksiä varten MQTT:ssä on ominaisuus, jolla voidaan tiedottaa asiakkaita asiakkaasta, johon katkesi odottamattomasti yhteys. Asiakas voi määrittää välittäjään ottaessa yhteyttä viimeisen tahdon ja testamentin, LWT (engl. Last Will and Testament). Odottamattomassa yhteyskatkossa välittäjä lähettää LWT:n eteenpäin muille asiakkaille. Tämä mekanismi sopii hyvin käyttötapauksiin, joissa toisten asiakkaiden on tarpeen tietää tietyn asiakkaan odottamattomasta yhteyskatkosta. [19,81]

5. PROTOKOLLAINTEGRAATION TOTEUTUS

Tässä luvussa esitellään kohdejärjestelmä, johon OPC UA:n ja valitun protokollan integraatio myöhemmin toteutetaan. Luvussa esitetään tiedonsiirrolle asetetut vaatimukset, jotka käytettävän integraation tulee täyttää toteutettavassa järjestelmässä. Vaatimukset pohjautuvat automaatio- ja IIoT-verkoille asetettuihin vaatimuksiin, joista kerrottiin tämän tutkimuksen luvussa 2. Lisäksi luvussa esitetään testaussovellus, jonka avulla valittua IoT-protokollaa vertaillaan S7-kommunikointiprotokollaan.

5.1 Kohdesovellus

Työn tilaajalla on kehitettävänä kaukolämpöjärjestelmän automaation tiedonkeruujärjestelmä. Tiedonkeruujärjestelmä käyttää OPC UA:ta palvelimien ja asiakassovellusten väliseen kommunikointiin. Se on valittu käytettäväksi, koska se mahdollistaa integroitavuuden ulkoisiin järjestelmiin, kuten eri valmistajien valvomojärjestelmiin. Kuvassa 22 esitetään kehitettävän järjestelmän keskeisimmät komponentit.



Kuva 22. *Kehitettävä tiedonkeruujärjestelmä. Kevyttä IoT-protokollaa käytetään tiedon siirtämisessä etäasemilta.*

Järjestelmällä kerätään automaatiojärjestelmän etäasemilta mittaus- tai tilatiedot käyttäen kevyttä kommunikointiprotokollaa. Tietovarastona toimii tietokanta. Lisäksi tiedonkeruujärjestelmä voidaan liittää valvomo-, tuotannon- tai toiminnanohjausjärjestelmiin OPC UA:ta käyttäen. OPC UA:ta ei kuitenkaan käytetä kommunikoinnissa etäasemien

ja tiedonkeruujärjestelmän välillä, koska sen tuki puuttuu vielä useista ohjelmoitavista logiikoista, erityisesti edullisista malleista. OPC UA:n monimutkaisen rakenteen takia protokollan toteuttaminen logiikkaan on työlästä ja haastavaa. Lisäksi ohjelmoitavien logiikkojen rajalliset resurssit rajoittavat toteuttamista. Etäasemien tiedonsiirto on toteutettu yleensä langattomilla radioilla, GPRS (engl. General Packet Radio Service), 3G- tai 4G-yhteyksillä. Tuolloin kaistanleveys voi olla pieni, jolloin se on käytettävä mahdollisimman tehokkaasti.

Koska tietoa siirretään internetissä, tiedonsiirto on oltava luotettavaa sekä tietoturvallista ja siksi tietoliikenne salataan. Keinot tietoturvan toteuttamiseen ovat esimerkiksi DTLS- tai TLS-salaustekniikat, riippuen käytettävästä protokollasta. Myös VPN-tunnelien käyttö on mahdollista ja se on vaihtoehtona, jos etäasemien ohjelmoitavat logiikat eivät tue salaustekniikoita. Tuolloin VPN-tunnelointi voidaan toteuttaa esimerkiksi sitä tukevilla modeemeilla. Jos VPN-tunneli luodaan modeemilla, salaaminen ei käytä ohjelmoitavan logiikan resursseja.

Edellä mainittujen asioiden lisäksi, tiedonkeruun kommunikointiin halutaan käytettäväksi avointa kommunikointiprotokollaa, joka on toteutettavissa edullisiin eri valmistajien logiikkoihin. Useat eri valmistajien logiikat tukevat omia valmistajakohtaisia protokollia, joiden standardit eivät ole avoimia. Koska protokollien standardit eivät ole avoimia, protokollan toteutus muun valmistajan laitteelle on haastavaa. Tällöin joudutaan käyttämään useita eri protokollia, jolloin järjestelmän arkkitehtuurista tulee monimutkainen ja ylläpidosta haastavaa. Testisovelluksessa käytetty S7-protokolla on myös valmistajakohtainen sekä suljettu protokolla. Sitä käytetään Siemensin ohjelmoitavissa logiikoissa sekä valvomo-ohjelmistoissa [82]. Lisäksi S7-protokolla käyttää kommunikoinnissaan pyyntö-vastaus-kommunikointimallia, jossa tiedonsiirto on hidasta, koska asiakas joutuu suorittamaan pyyntöjä vuorotellen ohjelmoitavilta logiikoilta.

5.1.1 Vaatimukset integraatiolle ja protokollalle

Toteutettavalle integraatiolle ja tiedonsiirrossa käytettävälle protokollalle asetettavat vaatimukset määriteltiin työn aloitusvaiheessa. Protokollan vaatimukset tulevat järjestelmälle asetetuista vaatimuksista ja ne pohjautuvat automaation tietoverkkojen, teollisuuden esineiden internetin asettamiin vaatimuksiin. Vaatimuksia käsiteltiin aliluvussa 2.3. Seuraavaksi käydään läpi asetetut vaatimukset.

1. **Tiedonsiirron tulee käyttää tiedonsiirtokapasiteetti tehokkaasti**

Sovelluksen käyttökohde on kaukolämpöjärjestelmien automaatio. Kaukolämpöjärjestelmät ovat hajautettuja ja kaukolämpöjärjestelmän etäasemat ovat fyysisesti hajallaan ja

kaukana toisistaan. Kustannussyistä tiedonsiirtoa ei toteuteta kaikkiin ala- tai mitta-asemiin kiintein yhteyksin. Tuolloin tieto siirretään langattomasti ja tiedonsiirtonopeus voi olla alhainen. Tästä syystä siirrettävä tietomäärä on minimoitava ja sanomakoot pidettävä pieninä. Siirrettävän mittaustiedon vaatimaan kaistanleveyteen ei voida vaikuttaa, joten käytettävän kommunikointiprotokollan ei tule muodostaa ylimääräistä tiedonsiirtoa. Lisäksi suuret sanomakoot ja sanomien lukuisat kentät kertovat protokollan monimutkaisesta rakenteesta. Tällöin protokollan toteuttaminen voi olla haasteellista resursseiltaan rajalliseen laitteeseen.

2. **Protokollan on sovelluttava järjestelmän arkkitehtuurin toteuttamiseen ja uusien etäasemien lisäys ei saa aiheuttaa kohtuuttomasti työtä**

Kaukolämpöverkot laajenevat eli järjestelmään tulee uusia automaatiolaitteita. Lisäksi vanhoja laitteita saatetaan uusia. Toteutettavan tiedonkeruujärjestelmän on oltava skaalautuva, eli uusien etäasemien lisäys on oltava mahdollista pienellä työpanoksella. Koska järjestelmän on oltava skaalautuva, tätä vaaditaan myös valittavalta protokollalta. Lisäksi valinnassa on huomioitava protokollan soveltuvuus järjestelmän arkkitehtuuriin, joka esitettiin kuvassa 22.

3. **Protokollan määrittelyn on oltava avoin ja mahdolliset väliohjelmistot lisenssimaksuttomia**

Avoin protokollan standardi mahdollistaa protokollan tuen toteuttamisen ohjelmoitaviin logiikkoihin pienellä kustannuksella. Jos protokollan arkkitehtuuri nojautuu väliohjelmistoihin, kehityksen pienen kustannustason säilyttäminen vaatii lisenssimaksuttomat tai avoimen lisenssin väliohjelmistot.

4. **Protokollalla on oltava tuki tietoturvalliselle tiedonsiirrolle**

Toteutettavassa järjestelmässä tietoa siirretään internetissä. Kerättävä tieto on luottamuksellista ja sitä ei haluta päästää ulkopuolisten haltuun. Tästä syystä valitulla protokollalla on oltava tuki tietoturvalliselle tiedonsiirrolle. Lisäksi jatkokehityshankkeeksi kaavaillaan prosessin optimointia, käyttäen kerättävää tietoa, jolloin järjestelmä osallistuu prosessin ohjaukseen. Tietoturva osaltaan takaa tiedon eheyden ja järjestelmän oikean toiminnan.

5. **Kaksisuuntaisen liikennöinnin mahdollisuus varmistettava**

Optimointi on tiedonkeruujärjestelmän kehityssuunta. Optimointijärjestelmällä olisi mahdollisuus käyttää OPC UA:ta tiedonsiirrossa tiedonkeruujärjestelmän kanssa. Jotta optimointia voidaan suorittaa kaukolämpöjärjestelmässä, täytyy tietoliikenteen olla kaksi-

suuntaista OPC UA-palvelimen ja etäasemien välillä. Tuolloin optimointijärjestelmä lähettää etäaseman ohjelmoitavaan logiikkaan esimerkiksi säätimien asetusarvoja sekä ohjauskomentoja.

6. **Protokollan on osaltaan tuotettava luotettava tiedonsiirto**

Järjestelmää kehitetään tällä hetkellä tiedonkeruuseen. Kehityksessä pidetään kuitenkin mielessä jatkokehitysmahdollisuudet ja liitettävyys erilaisiin valvomo-ohjelmistoihin, raportointi- tai optimointisovelluksiin. Tästä syystä tiedonsiirrolle on oltava määriteltävissä eri luotettavuustasoja. Esimerkiksi yksittäisen mittaustiedon katoaminen lähetyksessä ei ole vakavaa, mutta hälytystiedon saapumatta jäämisellä valvomoon voi olla pahemmat seuraukset. Kun järjestelmää kehitetään pidemmälle siten, että etäasemille voidaan lähettää asetusarvomuutoksia tai ohjauskomentoja, nämä täytyy lähettää tietyllä palvelunlaatutasolla. Asetusarvo täytyy toimittaa varmasti perille. Riippuen toteutustavasta, ei ole välttämättä haitallista, jos ohjelmoitava logiikka vastaanottaa saman asetusarvon useasti. Toisaalta ohjauskomentoa ei välttämättä saa toimittaa kuin täsmälleen kerran, mutta tämäkin riippuu siitä, kuinka logiikkaohjelma on toteutettu.

7. **Tiedon tapahtumajärjestys on varmistettava**

Tallennettuun tietoon täytyy sisällyttää tieto ajanhetkestä, jotta sillä voidaan muodostaa oikea kuva prosessin tilasta. Mittaustietojen mukana lähetetään asynkronisessa kommunikoinnissa lähetysajankohta tai järjestysnumero sekä mahdollisesti mittauksen laatutieto. Esimerkiksi OPC UA -muuttujat sisältävät valmiiksi attribuutin lähteen aikaleimalle, eli etäaseman ohjelmoitavan logiikan mittaus- tai tilatiedon näytteenottoajankohdalle. Sanomaan sisällytetty aikaleima asetetaan OPC UA -muuttujan lähteen aikaleimaksi.

8. **Toteutettavuus sekä käytettävyys edullisessa ohjelmoitavassa logiikassa**

Jotta voidaan todeta protokollan valinnan sekä integraation onnistuneen, on kevyen kommunikointiprotokollan tuki oltava mahdollista toteuttaa edulliseen ohjelmoitavaan logiikkaan. Jos toteuttaminen on mahdollista, hyödyntäen kohtuullisesti edullisen ohjelmoitavan logiikan muistia ja aiheuttamatta merkittävää ohjelman suoritusajan pidentymistä, voidaan ohjelmoitavan logiikan käyttöä ajatella myös etäaseman prosessin ohjauksessa. Tällöin kommunikoinnista ja prosessinohjauksesta saadaan selkeä kokonaisuus. Korkea resurssien käyttö ohjelmoitavasta logiikasta, sitoo laitteen vain kommunikointiin. Prosessinohjaus on pakko tuolloin toteuttaa erillisellä laitteella, jolloin järjestelmän ylläpidettävyys sekä käytettävyys kärsii. Lisäksi tästä muodostuu järjestelmään ylimääräistä kompleksisuutta sekä kustannuksia.

5.2 Protokollien vertailun tulokset

Tässä aliluvussa käydään läpi vaatimukset ja kuinka eri protokollat vastaavat esitettyihin vaatimuksiin. Vaatimukseen 8 vastataan luvussa 6. Protokollan tuen toteutettavuutta on haasteellista arvioida vain standardeihin pohjautuen. Vaatimukseen vastataan testausjärjestelmällä saaduilla tuloksilla. Testaukset suoritetaan protokollalla, joka valitaan vaatimuksiin 1–7 perustuen. Ensimmäisenä vertailusta esitetään taulukko, jotta kokonaiskuvan saaminen helpottuu. Taulukossa 2 käytetään merkintää (+), jos protokollan ominaisuudet vastaavat hyvin vaatimukseen tai (-), jos protokolla on ominaisuuksiltaan hyvin puutteellinen. Lisäksi taulukon solussa esitetään oranssi väri, jos vaatimukseen voidaan vastata, mutta vastaaminen teettää lisätyötä, määriteltävää tai jos vaatimuksen täyttämisessä on jokin epävarmuus. Tämän jälkeen esitetään valinnan perustelu protokollakohtaisesti. Vaatimukset olivat seuraavat:

1. Tiedonsiirron tulee käyttää tiedonsiirtokapasiteetti tehokkaasti.
2. Protokollan on sovelluttava toteutettavan järjestelmän arkkitehtuuriin ja uusien etäasemien lisäys ei saa aiheuttaa kohtuuttomasti työtä.
3. Protokollan määrittelyn on oltava avoin ja mahdolliset väliohjelmistot lisenssimaksuttomia.
4. Protokollalla on oltava tuki tietoturvalliselle tiedonsiirrolle.
5. Kaksisuuntaisen liikennöinnin mahdollisuus varmistettava.
6. Protokollan on osaltaan tuotettava luotettava tiedonsiirto.
7. Tiedon tapahtumajärjestys on varmistettava.

Taulukko 2. *Protokollavertailun tulokset. Vihreä väri ja (+)-merkintä tarkoittavat, että vaatimus täyttyy ja punainen (-)-merkinnällä tarkoittaa sitä, että vaatimus ei täyty. Oranssi väri tarkoittaa, että määrittelyyn voidaan vastata, mutta täyttäminen teettää lisätyötä, määriteltävää tai täyttämiseen liittyy epävarmuus.*

	1	2	3	4	5	6	7
AMQP	-	+	+	+	+	+	
HTTP	-		+	+	+	-	+
CoAP	+		+	+	+		-
MQTT	+	+	+	+	+	+	

AMQP

AMQP-protokollan version 1.0 otsikkokentän koko on 8 tavua. Lisäksi protokollassa siirretään kehysrungossa sanomasta tyyppikohtaista tietoa, jolloin sanomakoko kasvaa melko suureksi. Kun asiakasohjelma toteutetaan määrittelyn mukaisesti, täytyy asiakkaan tukea vähintään 512 tavun kehyskokoja. Tämä tila täytyy kiinteästi varata logiikan muistista.

Version 1.0 protokolla mahdollistaa julkaise-tilaa-mallin mukaisen kommunikoinnin, joten kommunikointi on pääpiirteittäin yksisuuntaista. Yksisuuntaisen ja tapahtumapohjaisen kommunikoinnin takia julkaise-tilaa-kommunikointimalli sopii hyvin tiedonkeruujärjestelmän arkkitehtuuriksi. Tuolloin tiedonkeruujärjestelmän ei tarvitse pyytää tietoja etäasemilta, vaan etäasemat lähettävät muuttuneen tiedon oma-aloitteisesti tiedonkeruujärjestelmään. Kun etäasemia on useita, tiedon pyytäminen vie aikaa ja tieto ei ole saatavilla reaaliaikaisesti, koska tieto täytyy pyytää etäasemilta vuorotellen.

AMQP-protokolla on erittäin monipuolinen ja se mahdollistaa useat kommunikointimallit. Toisaalta tästä syystä sen rakenne on monimutkainen. Kommunikointiosapuolet, eli säiliöt, muodostavat istuntoja, jotka sisältävät yksisuuntaisia linkkejä. Linkkejä voi olla istunnossa useita, joten kaksisuuntainen tiedonsiirto on toteutettavissa.

Istunnot ja linkit luodaan omilla komennoillaan. Monimutkaisesta rakenteesta kertoo myös protokollan kehyskoko. Monimutkaisen protokollan toteutus voi olla haasteellista ohjelmoitaviin logiikoihin, koska niiden ominaisuudet ovat rajallisia.

AMQP-protokolla on avoin ja useita ilmaisia väliohjelmistoja on olemassa. Protokollan versio 1.0 on muuttunut paljon versiosta 0.9.1. Automaation ohjelmistojen ja laitteiden elinkaari on pitkä, joten protokollan, jonka versioissa on suuria eroavaisuuksia, voi aiheuttaa ylläpidon kannalta haasteita. Suuret muutokset sitovat käyttämään protokollan versiota, jolle kommunikointi on alun perin toteutettu, koska muuttaminen on työlästä ja aiheuttaa kustannuksia.

Protokolla nojaa MQTT- ja HTTP-protokollien tavoin TLS-suojaukseen. Lisäksi määrittely kuvaa SASL-mekanismien käytön, jolloin määrittely kuvaa kuinka kehykset muodostetaan. Myös MQTT:n version 5.0 määrittely kuvaa SASL-mekanismien käytön. Sekä SASL- ja TLS-suojauksilla saadaan toteutettua asiakkaan autentikointi sekä tietoliikenteen salaus.

AMQP-protokollassa on helposti ymmärrettävät mekanismit sanomien välityksen luotettavuustasojen toteuttamiseen. Version 1.0 sanomien luotettavuustasojen toteutus nojaa merkkilippujen käyttämiseen, joilla kuitataan sanomat toimitetuiksi. Protokollalla voidaan toteuttaa järjestelmän vaatimat luotettavuustasot. Luotettavuustasot ovat soveltuvia

myös siihen tapaukseen, jossa järjestelmää jatkokehitetään siten, että etäasemille lähetetään ohjauksia tai asetusarvoja.

AMQP-protokollan, sekä MQTT-protokollan kehyksiin ei ole määritelty sisällytettäväksi aikaleimaa. Aikaleiman sisällytys sanomaan tulee toteuttaa erikseen. Ehdon täyttämiseen voidaan käyttää OPC UA:n PubSub-määrittelyä. PubSub-määrittely kuvaa, kuinka sen määrittelyn mukaisesti voidaan kommunikoida käyttäen AMQP- tai- MQTT-protokollaa. Määrittelystä voidaan hyödyntää rakennetta, jolla siirrettävä tieto, aikaleima tai laatutieto koostetaan AMQP- tai MQTT-sanomaan.

HTTP

HTTP 1.1 version otsikkokenttiä ei ole tarkasti määritelty, joten niiden vaatimaa kokoa ei saada ennen toteutusta. Otsikkokenttien koko todennäköisesti kasvaa suureksi, koska protokolla on tekstipohjainen. Lisäksi merkkijonojen käsittely ohjelmoitavissa logiikoissa käyttää runsaasti muistia. HTTP/2 protokolla on binäärinen ja otsikkokentän suuruus on 9 tavua, eli vertailtavista protokollista suurin. HTTP/2 määrittelyssä rajataan lisäksi sanomien maksimikoon vähimmäissuuruudeksi 16 384 tavua, jolloin määrittelyä tukevan laitteen pitää vähintään tukea tätä sanomakokoa.

Sekä HTTP 1.1 ja HTTP/2 toimivat asiakas-palvelin-arkkitehtuuria käyttäen, jossa asiakkaat toteuttavat pyyntöjä palvelimille. HTTP-protokollan käyttömahdollisuutta tutkittiin Siemensin S7-1200-sarjan ohjelmoitavien logiikkojen sisältämän web-palvelin ominaisuuden takia. Tuolloin asiakkaana on tiedonkeruujärjestelmä ja sen pitää suorittaa pyynnöt ohjelmoitavalle logiikalle.

Pyyntö-vastaus-kommunikointimallia käyttämällä tietoa siirretään turhaan, koska asiakas pyytää kaiken halutun tiedon. Lisäksi järjestelmässä on useita etäasemia eli palvelimia. Tuolloin tiedonkeruujärjestelmä suorittaa pyyntöjä etäasemilta vuorotellen. Tällöin tiedon saamisessa kuluu enemmän aikaa verrattuna tapahtumapohjaiseen sanomien toimitukseen. Etäasemien lisäämiseksi täytyy asiakasohjelmaa muokata, jotta se suorittaa pyynnöt lisätyiltä palvelimilta. Kaksisuuntaisen tiedonsiirron toteuttaminen kuitenkin onnistuu myös HTTP-protokollan asiakas-palvelin-arkkitehtuuria käyttäen. Tuolloin asiakas, eli tiedonkeruujärjestelmä, lähettää pyynnön resurssin muokkaamiseksi haluttuun arvoon.

HTTP-protokolla ei määrittelyssään kuvaa sanomien toimitukseen luotettavuutta takaavia mekanismeja. Vaatimuksen mukaiset mekanismit on toteutettava itse määriteltävällä tavalla. Itse suunnitellut ja kuvatut mekanismit lisäisivät kommunikoinnin toteutukseen työtä ja testattavaa.

Hyviä puolia HTTP-protokollassa on standardien julkaisuvuodet ja käyttölaajuus. Protokollan käyttöön on valmiita komponentteja ja useat alustat voivat kommunikoida HTTP-protokollalla. Protokollan otsikkoihin saa monipuolisesti tietoa, ja esimerkiksi aikaleiman sisällyttäminen otsikkoon on mahdollista. Lisäksi protokollassa on useita informatiivisia vastaustilatietoja, joiden perusteella voidaan reagoida eri tavoin ongelmatilanteissa. Sivun 29 taulukko 1 esitti vastaustilatietojen kategoriat.

CoAP

CoAP-protokolla on suunniteltu ominaisuuksiltaan rajoittuneisiin laitteisiin tai järjestelmiin, joiden tiedonsiirtoyhteydet ovat hitaita. CoAP-protokollan otsikkokentän koko on 4 tavua, mutta tietoliikenneprotokolla on UDP. UDP-tietoliikenneprotokolla mahdollistaa osaltaan pienen pakettikoon. UDP-protokollan takia CoAP-protokollan tietoturva toteutetaan käyttämällä DTLS-salausta. Se toteuttaa samankaltaisen salauksen kuin TLS-salaus.

Edellä mainittu pakettien pieni koko on tiedonkeruujärjestelmään hyvä ominaisuus, mutta UDP ei mahdollista luotettavaa tiedonsiirtoa. CoAP-protokollassa on tarvittavat mekanismit tiedonkeruuseen, mutta jatkokehityksen kannalta vaadittava täsmälleen kerran -luotettavuuden taso täytyy toteuttaa erikseen esimerkiksi hyödyntämällä sanoman tunnistetta. Lisäksi CoAP-protokollan uudelleenlähetyksessä on ruuhkanhallintaan mekanismi, jossa lisätään viive ennen uudelleenlähetystä. Ominaisuus eliminoi tietoliikenteen reaaliaikaisuuden. Reaaliaikaisuutta ei silti pidetä järjestelmässä omana vaatimuksenaan, mutta viiveettömyyttä voidaan pitää hyvänä ominaisuutena.

CoAP-protokolla toimii asiakas-palvelin-arkkitehtuurilla, jossa pyyntöjen toteutuksessa käytetään RESTful-arkkitehtuurityyliä. Vaikka protokolla toimii asiakas-palvelin-arkkitehtuurilla, voi sitä käyttävä laite toimia sekä palvelimena, että asiakkaana. Tämä mahdollistaa kaksisuuntaisen kommunikoinnin. CoAP-asiakas voi toimia järjestelmässä tarkkailijana, jolloin CoAP-protokollalla voidaan toimia asynkronisesti. Tällöin voidaan vähentää tietoverkon kuormitusta. Huonona puolena tarkkailijassa on se, että palvelimen täytyy muistaa resurssit, joita tarkkaillaan, sekä asiakkaat, jotka tarkkailevat. Tämä lisää laitteiden resurssien käyttöä.

CoAP-protokolla on määritelty vuonna 2014, eli protokolla on melko uusi. Uutuudestaan huolimatta se on saanut jonkin verran tilaa IoT-protokollana [9]. Lisäksi protokolla on yksinkertainen, mikä puoltaa toteutettavuutta ohjelmoitaviin logiikkoihin.

MQTT

Aiemmin todettiin, että julkaisija-tilaaja-arkkitehtuuria käyttävä malli mahdollistaa järjestelmän skaalautuvuuden, koska sanomien lähettäjät ja vastaanottajat ovat löyhästi kytkettyjä. Asiakas voidaan lisätä järjestelmään ilman, että muutoksia tarvitsee tehdä muihin asiakkaisiin. Lisäyksen ja aiheiden julkaisun jälkeen uudet tiedot ovat saatavissa. Asiakkaat voivat toimia sekä julkaisijoina että tilaajina, joten kommunikointi voidaan toteuttaa kaksisuuntaiseksi.

MQTT-protokolla valikoitui vertailussa ohjelmoitavassa logiikassa testattavaksi protokollaksi. Sen otsikkokenttä on pieni ja protokollan hyötykuorma on binäärinen. Huonona puolena on tekstipohjaiset sanomien aiheet, jotka lisäävät sanomakokoa. Toisaalta tekstipohjaisella aiheella voidaan kuljettaa hyötykuormasta hyödyllistä metatietoa, esimerkiksi mittauksien position. Versioon 5.0 tulleella alias-aiheella saadaan julkaisun aihe sisällytettyä kahteen tavuun. Ominaisuutta on hyvä käyttää usein lähetettävälle tiedolle. Protokollaan ei ole määritelty minimisanomakokoa, jota asiakaslaitteen on tuettava. Tuolloin asiakaslaitteella ei tarvitse varata tarpeettoman suurta muistia sanomien käsittelyyn.

MQTT-protokolla on avoin ja se julkaistiin alun perin vuonna 1999. Siihen tuli viimeisin päivitys huhtikuussa 2019. Protokollaan ei tullut suuria muutoksia ja sen ydintoiminnot pidettiin sellaisenaan. Parannuksia koki esimerkiksi protokollaa käyttävän järjestelmän skaalautuvuus, suorituskyky ja pienten asiakkaiden tuki, mikä edistää protokollan käyttöä esimerkiksi ohjelmoitavissa logiikoissa. MQTT-protokolla on suosittu IoT-käytössä ja sen käyttö on yleistynyt [9]. Tämä kertoo protokollan käytettävyydestä laitteissa, joissa resurssit ovat pienet.

MQTT vastaa hyvin esitettyihin vaatimuksiin. Protokolla tukee tarvittavat sanomienvaihdon luotettavuustasot, mutta lisäksi LWT ja pysyvät istunnot ovat järjestelmään käyttökelpoisia ominaisuuksia. LWT ominaisuutta käyttämällä voidaan ilmoittaa muita asiakkaita asiakkaasta, johon katkeaa yhteys odottamattomasti. Ominaisuudet ovat määritelty siten, että väliohjelmisto ylläpitää ja muistaa istuntotietoja. Tällöin ominaisuuksien toteuttaminen ei kuluta paljoa asiakaslaitteen muistia. Edellä mainittujen asioiden lisäksi MQTT:lle on olemassa useita ilmaisia väliohjelmistoja, jotka tukevat tarvittavia ominaisuuksia. AMQP-protokollan tavoin MQTT-sanomaan voidaan sisällyttää aikaleima Pub-Sub-määrittelyn mukaisesti.

5.3 Integraation toteutusvaihtoehdot

Tässä aliluvussa esitellään integraation toteutusvaihtoehdot. Integraatiosovelluksen tarkoitus on integroida OPC UA ja MQTT-protokolla sekä vertailuarvojen saamiseksi OPC UA ja S7-protokolla. Prototyypissä päädyttiin integroimaan MQTT-asiakas suoraan OPC UA -palvelimeen. Sovellusta voidaan käyttää tuolloin väliohjelmistona tai järjestelmän OPC UA -palvelimena. Toteutusvaihtoehtoina pidettiin protokollamuunninta, MQTT- ja OPC UA -asiakaskomponenttien integraatiota tai MQTT-asiakkaan integraatiota suoraan OPC UA -palvelimeen.

Protokollamuunnin

Protokollamuunnoksessa MQTT-asiakkaaseen, joka kommunikoi OPC UA-palvelimen kanssa, pitää toteuttaa OPC UA:n palvelupohjainen arkkitehtuuri. Toteutettavia palveluita tarvitaan useita pelkästään sivun 20 kuvan 11 OPC UA:n kommunikointipinon toteutukseen. Yhteyden luontiin käytetään kahden palvelujoukon palveluita (Discovery Service Set ja Open Channel Service Set). Palvelujoukot esitettiin sivun 18 kuvassa 9. OPC UA -palvelimen solmuja käsitellään Attribute-palvelujoukon palveluilla. Edellä mainittujen palvelujoukkojen toteuttaminen on tuolloin MQTT-asiakkaan minimivaatimus. Tässä tapauksessa OPC UA -palvelimeen ja MQTT-asiakkaaseen pitää määritellä valmiiksi solmut, ellei toteuteta myös NodeManagement-palvelujoukon palveluita solmujen luontiin ja poistoon. Ratkaisu johtaa monimutkaiseen asiakasohjelman luontiin, jonka toteutus ohjelmoitavaan logiikkaan on haasteellista.

Tätä toteutusvaihtoehtoa voidaan myös pitää tietoturvallisuuden kannalta haastavimpana verrattuna muihin toteutusvaihtoehtoihin. Tässä ratkaisuvaihtoehdossa haastavaksi muodostuu tietoturvallisen yhteyden luonti OPC UA -palvelimeen MQTT-protokollaa käyttäen. MQTT-asiakkaaseen tulisi luoda OPC UA:n salausmekanismit, ellei käytetä suojaamatonta yhteyttä. OPC UA:n salausmekanismien luonti ohjelmoitavaan logiikkaan olisi työlästä ja kuluttaisi ohjelmoitavan logiikan resursseja.

OPC UA -asiakkaan ja MQTT-asiakkaan integraatio

OPC UA -asiakkaan ja MQTT-asiakkaan integraatioon kuuluisi lisäksi OPC UA -palvelin. Tällä toteutustavalla järjestelmään tulee lisää kompleksisuutta erillisen OPC UA -asiakkaan takia. Toisaalta ratkaisu toisi muokattavuutta, koska OPC UA -palvelin on erillään. OPC UA -asiakkaan pitää toteuttaa edellisen toteutusvaihtoehdon tavoin useita OPC UA:n palveluita ja välittää tarvittavat tiedot MQTT-protokollalla. Kommunikointi etäaseman suuntaan olisi haastavaa. Palvelimen tulisi julkaista attribuuttien arvot OPC UA -asiakkaalle itsenäisesti, koska pyyntöjen toteuttaminen MQTT-protokollaa käyttäen on

raskasta. Tämän jälkeen OPC UA -asiakas suorittaa MQTT-julkaisun, jotta tieto saadaan lähetettyä ohjelmoitavalle logiikalle.

Tietoturvallisuuden kannalta ratkaisuvaihto olisi parempi kuin muut. Jos MQTT-asiakas integroidaan OPC UA -asiakkaaseen, OPC UA -asiakas ottaa yhteyden OPC UA -palvelimeen. Useissa OPC UA -asiakassovelluksissa tietoturvallinen kommunikointi kuuluu perusominaisuuksiin. OPC UA -asiakkaan saisi muodostamaan salatun yhteyden OPC UA -palvelimeen. Ohjelmoitava logiikka MQTT-asiakkaana voi käyttää TLS-salausmekanismia, autentikointia ja auktorisointia. Tässäkin ratkaisuvaihtoehdossa on kiinnitettävä erityistä huomiota välittäjän turvallisuuteen. Jos välittäjään pääsee julkaisemaan tuntematon taho, tieto päätyy OPC UA -palvelimelle.

OPC UA -palvelimen ja MQTT-asiakkaan integraatio

Tätä ratkaisutapaa päädyttiin käyttämään prototyypissä. OPC UA -palvelimen ja MQTT-asiakkaan integraatiossa järjestelmään voi kuulua lisäksi erillinen OPC UA -palvelin tai integraation palvelinta voidaan käyttää suoraan järjestelmän palvelimena. Jos toteutettavassa järjestelmässä ei käytetä erillistä OPC UA -palvelinta, täytyy toteuttaa MQTT-aiheiden käsittelyyn hallintaohjelma. Tällä tavoin vältytään OPC UA -palvelimen pysäytykseltä muutoksia tehdessä.

Ratkaisun hyvä puoli on, että ei tarvitse erikseen toteuttaa OPC UA:n lukuisia palveluita, sillä vaadittavat toiminnot voidaan toteuttaa suoraan palvelinohjelmaan. Lisäksi kaksisuuntainen tiedonsiirto pystytään toteuttamaan reaaliaikaisesti. Kun OPC UA -palvelimen muuttujan arvoa muutetaan, suoritetaan MQTT-asiakasta käyttäen julkaisu. Jos käytetään OPC UA -asiakasta integraatiossa, joutuu asiakas suorittamaan kiertokyselyn OPC UA -palvelimelle ja vasta tämän jälkeen voidaan sanoma välittää MQTT-protokollaa käyttäen. Tästä syntyy viivettä.

Tällä ratkaisutavalla tietoturvallisuudessa täytyy ottaa erityisesti huomioon välittäjän ja MQTT-asiakkaan ja OCP UA -palvelimen integraation yhteys. Jos ulkopuolinen ohjelmisto pääsee ottamaan yhteyden välittäjään ja voi julkaista sallittuja aiheita, päätyy julkaisut OPC UA -palvelimelle. Tästä syystä välittäjään täytyy ottaa käyttöön autentikointi ja auktorisointi, jotta vain tunnetut asiakkaat voivat ottaa yhteyttä. Lisäksi yhteydet on oltava salattuja, jotta käyttäjätunnuksia ja salasanoja ei saa haltuun liikennettä kuuntelemalla. MQTT-protokollaa käytettäessä ja tietoturvallisuutta luodessa, välittäjä täytyy ottaa huomioon.

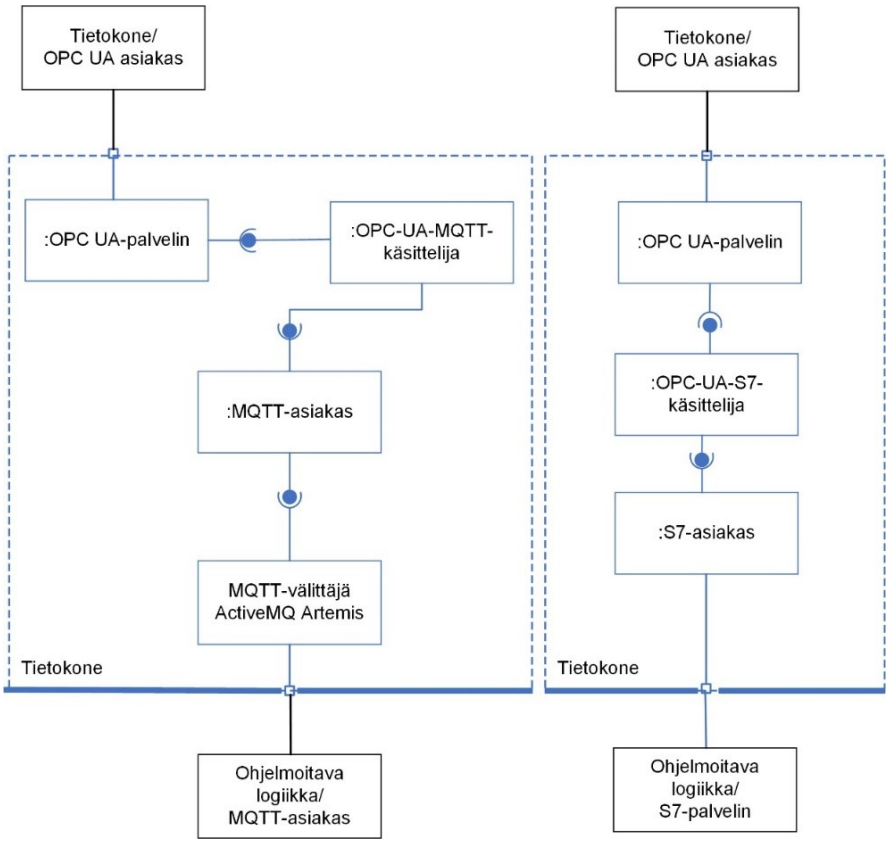
5.4 Testausjärjestelmä

Integraatiosta toteutettiin prototyyppi ja sitä käytettiin testausjärjestelmässä, jolla testattiin vertailussa valituksi tullutta MQTT-protokollaa ja vertailtiin sitä S7-protokollaan. Toteutuksen tarkoitus on testata protokollan suorituskykyä ja käytettävyyttä ohjelmoitavissa logiikoissa. Lisäksi sovelluksella saadaan kokemusta mahdollisesta järjestelmäarkkitehtuurista OPC UA:n ja protokollan integraatiossa. Saatua kokemusta voidaan hyödyntää kehitettävässä tiedonkeruujärjestelmässä, mutta testauksen prototyyppiä ei ole tarkoitus käyttää sellaisenaan toteutettavassa järjestelmässä. Toteutettavan järjestelmän OPC UA -palvelin, MQTT-asiakasohjelmistot ja välittäjät voivat olla muita, kuin testauksessa käytetyt, koska toteutettavan järjestelmän ominaisuudet ja vaatimukset tarkentuvat myöhemmin kehityksessä. Tästä syystä prototyyppiin ei myöskään kehitetty käyttöliittymää ja erillistä aiheiden ja OPC UA -muuttujien hallintaohjelmistoa. Kuitenkin logiikkaohjelmaan toteutettiin kattavasti MQTT-asiakkaan ominaisuudet, jotta pystyttiin arvioimaan resurssien tarve mahdollisimman tarkasti.

Testaussovelluksesta toteutettiin kaksi versiota, koska vertailtiin kahta eri arkkitehtuuriin ja kommunikointimalliin perustuvaa protokollaa. Testijärjestelmä koostuu seuraavista pääosista: ohjelmoitavasta logiikasta, joka toimii MQTT-asiakkaana tai S7-palvelimena, integraatiosovelluksesta ja OPC UA -asiakkaasta. Aliluvussa 5.4.1 esitetään testattavien protokollien integraatio OPC UA -palvelimeen, aliluvussa 5.4.2 testauksessa käytetty väliohjelmisto ja aliluvussa 5.4.3 esitetään ohjelmoitavan logiikan MQTT-asiakkaan toteutus.

5.4.1 Integraatiosovellus

Sovellus, jossa integroidaan MQTT- tai S7-kommunikointiprotokollat OPC UA -palvelimeen, koostuvat OPC UA -palvelinkomponentista, MQTT- tai S7-asiakaskomponentista sekä sanomien käsittelykomponentista. Kun käytössä on MQTT-protokolla, järjestelmään kuuluu lisäksi MQTT-välittäjä, joka voi olla käytössä samalla tai eri laitteella. MQTT-asiakaskomponentti muodostaa yhteyden välittäjään. Kuva 23 esittää molempien sovellusten rakennetta. Komponentit ovat toteutettu Node.js suoritusympäristöön ja kirjoitettu JavaScript-kielellä.



Kuva 23. *Vasemmalla testausjärjestelmän rakenne, kun käytetään MQTT-protokollaa ja oikealla, kun käytetään S7-protokollaa.*

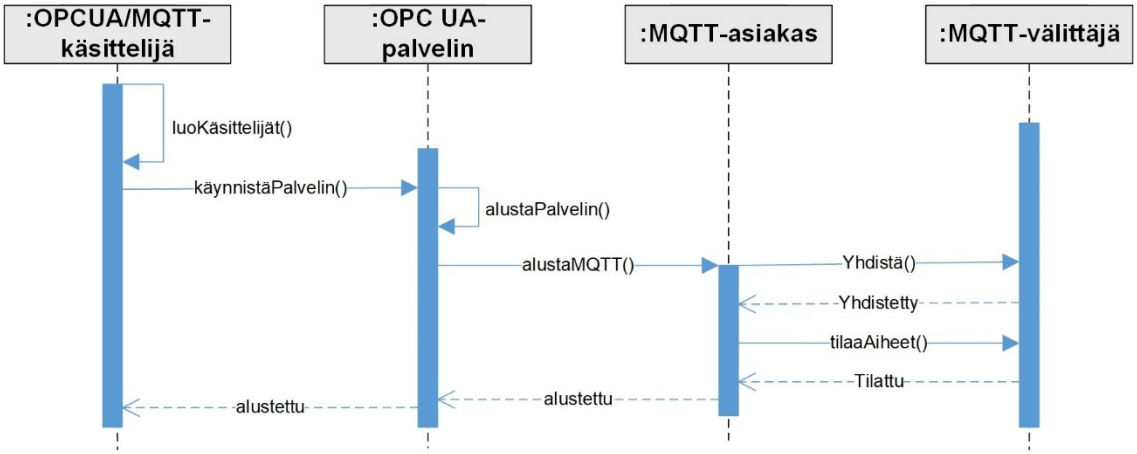
Testin OPC UA -palvelimena toimi Node.js ympäristöön toteutettu MIT-lisensoitu (engl. Massachusetts Institute of Technology) palvelin, joka hyödyntää Node.js ympäristön asynkronisuutta. Käytetystä OPC UA -pinosta löytyy useita demosovelluksia asiakas- ja palvelinohjelmien luontiin. [83] MIT-ohjelmistolisenssi antaa käyttö-, kopiointi- ja julkaisuoikeiden ohjelmistoon ja sen dokumentaatioon. MIT-lisensoituja ohjelmistoja voidaan käyttää myös kaupallisesti. [84]

Integraation MQTT-asiakas on Node.js MQTT-kirjastokomponenteista toteutettu komponentti. MQTT-asiakaskomponentit ovat myös MIT-lisensoitu. Komponentit ovat kirjoitettu JavaScript-kielellä, mutta tuki TypeScriptille on olemassa. MQTT-asiakaskomponentit tukevat MQTT:n versiota 3.1.1. Julkaisijalla on kehityksessä version 5.0 tuki. Asiakaskomponentit tukevat muun muassa MQTT:n palvelunlaatutasoja, autentikoinnin, viimeisen toiveen ja testamentin sekä TLS-suojatun kommunikoinnin. [85]

Myös S7-protokollalle löytyvät Node.js ohjelmistokirjastot. Ohjelmakirjaston komponenteilla voidaan toteuttaa kommunikointi Siemensin S7-300/S7-400 sekä S7-1200/S7-1500 -sarjojen ohjelmoitaviin logiikkoihin. [86] Komponenteilla toteutettiin kuvan 23 testausjärjestelmän S7-asiakkaan osuus.

OPC UA/MQTT-integraatiosovelluksen toiminta

Sovelluksen eteneminen käynnistyksessä esitetään kuvassa 24. Käynnistettäessä, kun käytetään MQTT-protokollaa, OPCUA/MQTT-käsittelijä suorittaa *luoKäsittelijät*-funktion. Käsittelijöitä määritellään niille OPC UA -muuttujille, joiden arvoja on tarkoitus lähettää tai vastaanottaa käyttäen MQTT-protokollaa. Käsittelijät sisältävät tiedot muuttujan tietotyypistä, muuttujan arvon sekä funktion sanoman hyötykuorman käsittelyyn. Tällä tavoin mahdollistetaan useat eri tietotyypit. Käsittelijöitä voidaan luoda niille OPC UA:n tietotyypeille, joille löytyvät vastineet ohjelmoitavista logiikoista. Lisäksi käsittelijöillä voidaan rajata, mitkä OPC UA -muuttujien muutokset lähetetään MQTT-välittäjälle tai mitä välittäjälle tulevia sanomia ei välitetä OPC UA -palvelimelle. Muuttujille, joiden arvo lähetetään MQTT-välittäjälle OPC UA -palvelimelta, asetellaan lisäksi käsittelijälle julkaisun palvelunlaatutaso 0–2. Ominaisuus on tarpeellinen, jos välittäjä sijaitsee eri tietokoneella kuin integraatiosovellus.

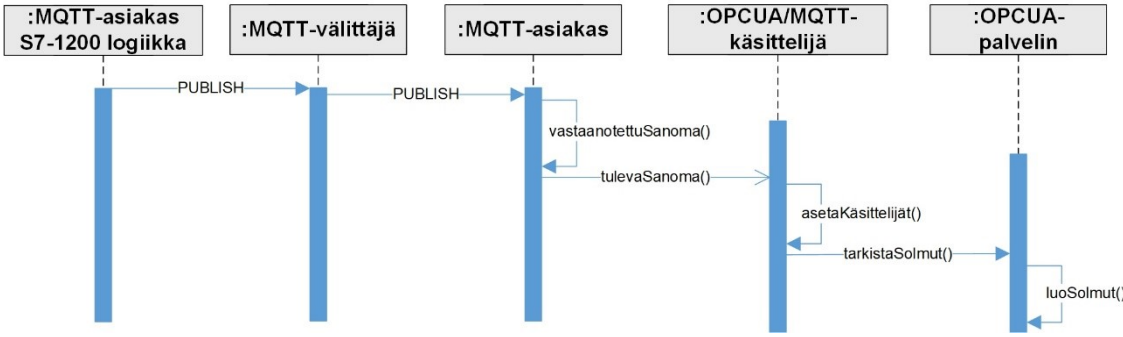


Kuva 24. *Integraatiosovelluksen käynnistys käytettäessä MQTT-protokollaa.*

Käsittelijöiden luonnin jälkeen sovellus alustaa OPC UA -palvelimesta instanssin. Onnistuneen alustuksen jälkeen palvelin alustaa MQTT-asiakkaan, joka yhdistää välittäjään. Kun yhteys välittäjään on muodostettu, MQTT-asiakas tilaa esimääritellyt aiheet sanomavälittäjältä. Järjestelmässä käytetystä sanomavälittäjästä kerrotaan aliluvussa 5.4.2. Käsittelijät, OPC UA -palvelimen sekä MQTT-asiakkaan konfiguraatioasetukset ja tilattavat aiheet annetaan käynnistyksessä parametreina. Tilattaville aiheille asetellaan myös palvelunlaatutaso käynnistyksessä. Integraatiosovelluksen käynnistyminen ei ole sidoksissa ohjelmoitavaan logiikkaan, joten ohjelmoitava logiikka voi ottaa yhteyden välittäjään eri aikaan.

Tässä testaukseen toteutetussa prototyypissä ei käytetty salausta, mutta kehitettävään toteutukseen otetaan käyttöön autentikointi, TLS-salaus ja varmenteet. Kun käytetään salattua yhteyttä, käytetään sertifikaattia. Kehitettävään toteutukseen välittäjässä on

otettava käyttöön autentikointi, jotta asiakkaat saavat muodostettua yhteyden vain salasanalla ja käyttäjätunnuksella. Sertifikaatti, käyttäjätunnus ja salasana annetaan *Yhdistä*-funktion parametreiksi. Lisäksi otetaan käyttöön valtuutukset, joilla voidaan sallia asiakkaiden tiettyjen aiheiden julkaisu tai tilaus. Tämä asetellaan välittäjään. Näillä menetelmillä asiakas voi varmentaa palvelimen sekä palvelin voi varmentaa yhteyttä ottavat asiakkaat. Lisäksi TLS-salauksen käyttäminen tuottaa tiedon eheyden ja luottamuksellisuuden varmistamisen.



Kuva 25. *Saapuneen MQTT-sanoman, palvelunlaatutasolla QoS=0, käsittely sovelluksessa.*

MQTT-sanoman vastaanoton käsittely esitetään kuvassa 25. Ennen kuin logiikka voi suorittaa julkaisun, sen on otettava yhteys välittäjään (toiminta kuvattu aliluvussa 5.4.3). Kun ohjelmoitava logiikka suorittaa julkaisun välittäjälle, välittäjä välittää julkaisun sovellukselle. Tuolloin sovellus kutsuu MQTT-asiakaskomponentin *vastaanotettuSanoma*-funktiota, jolla suoritetaan käsittelijäkomponentin sanoman käsittelyfunktio *tulevaSanoma*. Funktio saa parametreina sanoman aiheen ja hyötykuorman. Aluksi funktio tarkistaa onko käsittelijä määritelty aiheelle. Jos käsittelijä on määritelty, funktio kutsuu aihetta vastaavan käsittelijän funktiolla *asetaKäsittelijät.*

Funktio käsittelee vastaanotetun MQTT-sanoman hyötykuorman alustuksessa määritellyn tietotyypin mukaisesti ja käsittelijän säilömä muuttujan arvo asetetaan. Tämän jälkeen tarkistetaan, onko OPC UA -palvelimen osoiteavaruuteen muodostettu aihetta vastaavaa solmua sekä OPC UA -muuttujaa *tarkistaSolmut*-funktiolla. Jos solmua ei ole, esimerkiksi silloin kun aihe vastaanotetaan ensimmäistä kertaa, funktio muodostaa palvelimelle kansiorakenteen käyttäen OPC UA:n kansiotyyppejä (engl. FolderType) sekä luo OPC UA -muuttujan. Tähän käytetään *luoSolmut*-funktiota, jossa kansiorakenteen luontiin sekä muuttujien lisäyksiin käytetään OPC UA -palvelimen funktioita. Muuttujan tietotyyppi määräytyy käsittelijän tietotyypin mukaiseksi. Kansiorakenne muodostetaan aiheessa käytettyjen ”/” -merkkien perusteella. Merkkijonotyyppinen solmutunniste muodostetaan aiheen loppuosan, eli viimeisen ”/”-merkin jälkeisen tekstin perusteella ja

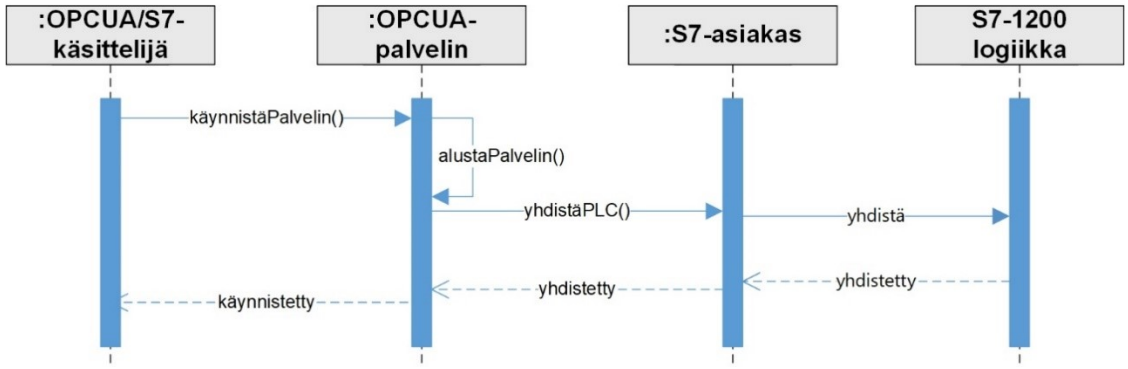
solmu lisätään osoiteavaruuden kansiorakenteeseen. Jos käsittelijää ei ole määritelty, luodaan ilmoitus lokiin ja sanomaa ei käsitellä.

Sanoman vastaanotossa käsittelijä säilöö muuttujan arvon. Kun OPC UA -asiakas suorittaa attribuutin lukupyynnön, esimerkiksi attribuutti-palvelujoukosta, OPC UA -palvelimelle, palvelin palauttaa käsittelijän säilömän muuttujan arvon asiakkaalle sekä muuttujan OPC UA -tilatiedon (engl. StatusCode).

Myös OPC UA -palvelimella suoritettava MQTT-julkaisu toteutettiin. Kun OPC UA -asiakas muuttaa OPC UA -palvelimen muuttujan, jolle käsittelijä on määritelty, arvoa kirjoitapyynnöllä, suoritetaan aiheen julkaisu. Aihe muodostetaan samoin kuin sanoman vastaanotossa. Muuttunut arvo asetetaan sanoman hyötykuormaksi, jonka MQTT-asiakas lähettää välittäjälle. Onnistuessaan OPC UA -palvelin palauttaa tilatiedon OPC UA -asiakkaalle. Epäonnistuessaan palautetaan kommunikointivirhetilatieto.

OPC UA/S7-protokollan integraatiosovelluksen toiminta

Testisovelluksen S7 osuus toteutettiin vain vertailua varten, joten esimerkiksi sovelluksen laajennettavuutta ei mietitty. Käytettäessä S7-protokollaa testisovelluksessa ei käytetä käsittelijöitä. Kuva 26 esittää sovelluksen käynnistymisen ja yhteyden luonnin ohjelmoitavaan logiikkaan.



Kuva 26. *Sovelluksen käynnistys, kun käytössä on S7-protokolla.*

Yhteyden luonnissa määritetään ohjelmoitavalta logiikalta pyydettävät muuttujat sekä alustetaan palvelimelle vastaavat OPC UA -solmut. Aluksi ohjelma alustaa OPC UA -palvelimen *alustaPalvelin*-funktiolla samoin kuin käytettäessä MQTT-protokollaa. Ohjelmakoodiin on alustettu ohjelmoitavan logiikan IP-osoite sekä prosessorin korttipaikan numero. Tämän jälkeen palvelin pyytää *yhdistäPLC*-funktiolla S7-asiakasta yhdistämään ohjelmoitavaan logiikkaan. Funktio saa parametreina määritellyt yhteysasetukset, eli IP-osoitteet ja korttipaikan numeron, joita S7-asiakas tarvitsee kommunikoidessa ohjelmoitavan logiikan kanssa.

Koska S7-protokolla käyttää kommunikoinnissaan asiakas-palvelinkommunikointimallia, pyydetään logiikalta muuttujien arvoa vasta silloin, kun OPC UA -asiakas pyytää palvelimelta OPC UA -muuttujien arvoa. Kun S7-asiakas saa vastauksen ohjelmoitavalta logiikalta, asetetaan arvo OPC UA -muuttujaan. Edellä kuvattu periaate toteutettiin protokollien vertailun suorittamiseksi.

5.4.2 Väliohjelmisto

Testausjärjestelmän väliohjelmistona toimi avoimen lähdekoodin, Apache 2.0 lisensoitu, sanomanvälittäjä Apache ActiveMQ Artemis 2.8.1. Se on asynkroninen sanomapohjainen väliohjelmisto, joka mahdollistaa erilaisten järjestelmien löyhän kytkennän. Artemis tukee pisteestä-pisteeseen sekä julkaise-tilaa-mallin mukaisia kommunikointimalleja useilla eri sanomien välittämisen luotettavuustasoilla. [87]

Apachen väliohjelmiston ydin koostuu yksinkertaisista Java-olioista, POJO (engl. Plain Old Java Object). Jokaisella palvelimella on oma nopean suorituskyvyn loki, johon palvelin säilöö sanomat ja muut säilytettävät tiedot. Myös tietokannat ovat tuettuja käyttäen JDBC-rajapintaa (engl. Java Database Connectivity), mutta tämä ei ole oletuksena käytössä huonomman suorituskyvyn takia. [87]

Apache ActiveMQ Artemis tukee useita protokollia kuten MQTT-protokollan versiota 3.1.1 ja AMQP-protokollan versiota 1.0. Lisäksi se tukee sanomienvaihdon ohjelmointirajapintoja sekä RESTful API -rajapintoja. [87]

Windows-, Linux- ja Unix-käyttöjärjestelmät tukevat Apache ActiveMQ Artemis-välittäjää ja sitä voidaan suorittaa niissä usealla eri tavalla. Tässä tutkimuksessa välittäjää käytettiin Windows NT-palveluna, jolloin välittäjä käyttää suorittamiseen Java Service kääreitä. Muita suoritustapoja ovat esimerkiksi Unix-palvelu tai välittäjän sulauttaminen JMS-välittäjäksi (engl. Java Message Service). [88]

Apache ActiveMQ Artemis tukee vaadittavia ominaisuuksia, kuten MQTT-protokollan palvelunlaatuja. Se valittiin testin väliohjelmistoksi muun muassa web-käyttöliittymän takia. Käyttöliittymällä voidaan tehdä sanomienvaihdon perustoimintoja, kuten tehdä julkaisuja tai hallita tilauksia. Lisäksi käyttöliittymässä on mahdollista monitoroida yhteydessä olevia asiakkaita, ja seuraamaan asiakkaiden aktiivisuutta. [87] Ominaisuudet koettiin hyödylliseksi testattaessa ohjelmoitavan logiikan MQTT-asiakkaan toimintoja.

Apache ActiveMQ Artemis välittäjä konfiguroidaan XML-tiedostolla. Oletustiedostosta poistettiin ylimääräiset protokollahyväksyjät (engl. acceptors) ja jätetty hyväksyjä MQTT-protokollalle. Tuolloin välittäjä ei käytä muita protokollia, eikä salli yhteydenottoja muilla

protokollilla. Tällä pienennettiin välittäjän kuormitusta ja selkeytetään web-käyttöliittymää. Yhteydenotot hyväksyttiin vain käyttämällä porttia 1883.

5.4.3 Siemens S7-1200 MQTT -asiakas

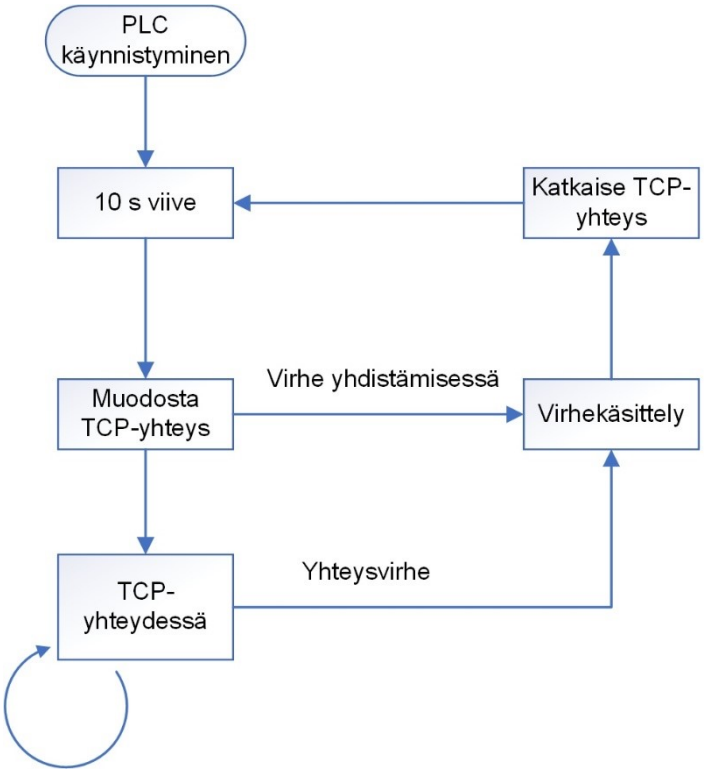
Järjestelmän käyttökohde on kaukolämpöverkkojen automaatio, jossa tietojen keräämiseen etäasemilta käytetään edullisia ohjelmoitavia logiikkoja. Järjestelmän testauksessa käytettiin Siemensin S7-1215C DC/DC/DC, 6ES7 215-1AG40-0XB0, ohjelmoitavaa logiikkaa. Logiikan laiteohjelmiston versio oli 4.1. S7-1200-sarjan logiikat ovat rakenteeltaan modulaarisia ja niihin on integroitu mallikohtaisesti tuloja ja lähtöjä. Lisäksi S7-1200-sarjan logiikat sisältävät integroidut Ethernet-rajapinnat, joita voidaan käyttää TCP-kommunikointiin. [89] Nämä ominaisuuden mahdollistavat osaltaan sen, että logiikkaa voidaan käyttää toteutettavan järjestelmän etäasemien tiedonsiirron toteutukseen. Tämän ohjelmoitavan logiikan puute tällä ohjelmistoversiolla, tämän työn kannalta, oli TLS-salauksen puute. Ominaisuus tuli laiteohjelmiston versioon 4.3.

MQTT-asiakkaan toteutuksessa käytettiin pohjana internetistä ladattavissa olevaa MQTT-julkaisijan versiota 1.1 [90]. Asiakasohjelmaan oli toteutettu vain sanomien julkaisu, joten ohjelmakirjaston MQTT-asiakaskomponenttiin toteutettiin aiheiden tilaamisen ja tilauksien purkamisen toteuttavat toiminnot.

MQTT-asiakkaan julkaisijan ominaisuudet olivat kattavat. Yhteyden muodostuksessa välittäjään oli mahdollista määritellä autentikointi, eli käyttäjätunnus ja salasana. Lisäksi ohjelmaan oli mahdollista määritellä LWT-sanoma, jonka välittäjä välittää muille odottamattomassa yhteyden katkeamisessa. Sanomat voitiin julkaista eri laatutasoilla sekä julkaisut voitiin määritellä myös pysyviksi. Mittauksien ja tilatietojen julkaisemiseksi luotiin yksinkertainen julkaisumenetelmä. Mittaukset julkaistaan, kun mittausarvo on muuttunut määritellyn kynnysarvon verran. Tilatiedot lähetettiin muutoksesta. Mittaus- tai tilatiedot tunnistettiin aiheella.

MQTT-asiakkaaseen toteutettiin aiheiden tilaaminen (SUBSCRIBE) ja tilauksien purkaminen (UNSUBSCRIBE). Toiminnot toteutettiin MQTT:n version 3.1.1 määrittelyllä, koska asiakassovellus perustui tähän määrittelyyn. Asiakasohjelmalla voidaan tilata aiheita, joiden sanomien sisältö mahtuu 254 tavuun. Aiheen merkkien määrä rajattiin 30. Tilaukseen voidaan määritellä kaikki määrittelyn mukaiset palvelunlaatutasot 0–2. Logiikkaohjelmaan toteutettiin lisäksi tilauksien purkamisen toiminnot. Koska tilauksille voidaan asettaa palvelunlaatutasot 1 ja 2, piti asiakkaaseen toteuttaa välittäjän julkaisuille kuittaussanomat. Asiakkaaseen toteutettiin PUBACK palvelunlaatutason 1 kuittaami-

seen ja PUBREC sekä PUBCOMP palvelunlaatutason 2 julkaisun vastaanoton suorittamiseksi. Lisäksi asiakasohjelmaan toteutettiin toipumismenettelyt yhteyskatkosta sekä virtakatkosta, koska näitä ei ollut toteutettu ladattavissa olevaan MQTT-asiakkaaseen.

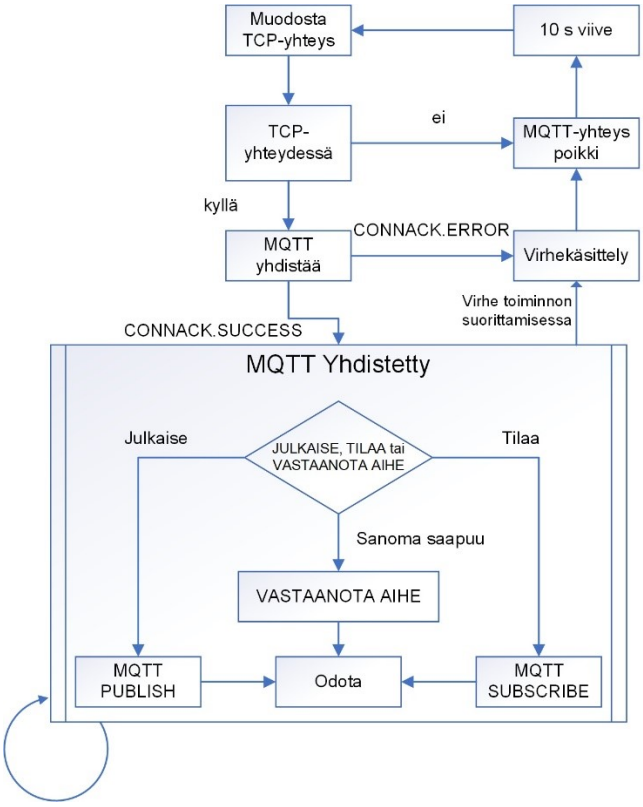


Kuva 27. *S7-1200 MQTT-asiakasohjelman TCP-yhteyden muodostuksen tilakaavio. Muokattu lähteestä [90].*

Kuva 27 esittää ohjelmoitavan logiikan yhteyden muodostusta MQTT-välittäjään. Ensimmäisessä ohjelmakierrossa logiikassa käynnistetään ajastin. 10 sekunnin jälkeen logiikka aloittaa TCP-yhteyden muodostamisen ohjelmassa määritettyyn IP-osoitteeseen sekä porttiin. Logiikkaohjelman TCP-kommunikoinnin lähetykseen liittyvät toiminnot toteutetaan Siemensin valmiilla TSEND_C- ja vastaanotto TRCV_C-ohjelmalohkoilla. Käyttäessä edellä mainittuja ohjelmalohkoja, yhteyden muodostamisen jälkeen logiikka hallinnoi TCP-yhteyden ylläpitoa ja tarkkailua [89].

Yhteyden katketessa odottamattomasti, tai jos yhteyden muodostaminen ei onnistu, suoritetaan virheenkäsittely, jossa logiikka palauttaa virhetiedot, josta näkee TSEND_C-ohjelmalohkon palauttaman diagnostiikkatiedon. Lisäksi logiikan kommunikointilohkot suorittavat TCP-yhteyden katkaisun toiminnot TSEND_C-ohjelmalohkojen toiminnoilla. Tämän jälkeen ohjelma asettaa MQTT-ohjelman tilaksi MQTT-yhteys poikki ja ohjelma siirtyy odottamaan uutta yhteydenottoyritystä. Uutta yhteydenottoa yritetään 10 sekunnin kuluttua.

Kun TCP-yhteys muodostetaan onnistuneesti, aloitetaan MQTT-protokollan mukainen yhteyden muodostaminen. Kuvassa 28 esitetään MQTT-protokollan toimintojen suoritus, kun ohjelmoitava logiikka on ottanut TCP-yhteyden MQTT-välittäjään.



Kuva 28. *MQTT yhdistäminen ja asiakasohjelman toimintojen suoritus ohjelmoitavassa logiikassa. Muokattu lähteestä [90].*

Kun MQTT-yhteys on avattu, ohjelma siirtyy odottamaan komentoja tai sanoman saapumista. Jos MQTT-yhteyden muodostuksessa tapahtuu virhe, uutta yhteydenmuodostusta yritetään 10 sekunnin kuluttua.

Jos virhe tapahtuu suoritettaessa julkaisua, tilausta tai tilauksen purkua, ohjelma palauttaa virhetiedon, joka ilmaisee mitä MQTT-toimintoa suoritettaessa virhe tapahtui. Tämän jälkeen ohjelma katkaisee MQTT-yhteyden MQTT-protokollan mukaisella DISCONNECT-toiminnolla. Uusi yhteys muodostetaan 10 sekunnin kuluttua katkaisusta. Virhetietojen palautus oli toteutettu ohjelmassa yhdistämisen ja julkaisujen toiminnoille, joten virhetiedot lisättiin myös tilauksien tekemiselle, tilauksien purulle ja vastaanotolle. Virhetiedot ovat tarkasteltavissa kehitysohjelmassa.

Testaussovelluksessa onnistuneen MQTT-yhteyden luonnin jälkeen logiikkaohjelma tilaa ennalta määritellyt aiheet halutulla palvelunlaatutasolla 0–2. Kun tilattu sanoma saapuu, sijoitetaan aihe sekä hyötykuorma merkkijonoon, joiden pituudet ovat rajoitettu 254 merkkiin. Onnistuneen aiheiden tilaamisen jälkeen, ohjelma aloittaa ennalta määriteltyjen julkaisujen tekemisen halutuilla palvelunlaatutasoilla 0–2.

Julkaisuja varten ohjelmassa on määriteltävissä julkaisun aihe ja hyötykuorma merkkijonoina. Aiheen merkkien määrä on rajoitettu 30:een ja hyötykuorman koko rajautuu 254 merkkiin. Yksi merkki käyttää yhden tavun muistia. Hyötykuorman koko rajoittuu 254 merkkiin, koska S7-1200-sarjan logiikan merkkijonot ovat enimmillään 254 merkkisiä. Aiheen merkkijonot ovat rajoitettu muistin säästämisen takia. Toiminnon suorituksen jälkeen ohjelma palaa odottamaan komentoja tai sanomien saapumista.

5.5 Suorituskyvyn testaus

Protokollan suorituskykyä arvioidaan testausympäristön avulla kolmella indikaattorilla: siirrettävien kehyksien kokonaissuuruudella, tiedon käsittelyyn kuluvalla ajalla sekä ohjelmoitavan logiikan suorituskykytiedoilla. Siirrettävä kokonaistietomäärä vaikuttaa tietoliikenneyhteyden kaistanleveyden käyttöön. Protokollan käsittelyyn ja lähetykseen kuluvalla ajalla sekä muistin käytöllä voidaan arvioida protokollan käytettävyyttä automaatiojärjestelmässä. Protokollaa verrataan S7-protokollalla saatuihin suorituskykyarvoihin, jotta tiedetään, mitä tulokset tarkoittavat. Ilman vertailtavia arvoja on haasteellista arvioida protokollan toteutuksen suorituskykyä ja hyvyyttä. Testausjärjestelmän OPC UA -palvelin ja testattavien protokollien integraatiosovellus sekä välittäjäohjelmisto olivat samalla tietokoneella. Tietokone oli samassa lähiverkossa ohjelmoitavan logiikan kanssa, jotta tietoverkko ei aiheuttanut juurikaan viivettä.

5.5.1 Siirrettävä tietomäärä

Siirrettävät tietomäärät mitattiin MQTT- ja S7-protokollalla. Molemmilla protokollilla siirrettiin samanlaiset hyötykuormat sisältävät sanomat. Hyötykuormat olivat 100 tavun suuruisia. Hyötykuorman lisäksi siirrettäviä tietoja ovat protokollien otsikot ja parametritiedot. Lisäksi MQTT-protokollalla hyötykuormaan lisätään 20-merkkinen aihe. Siirrettävä kokonaistietomäärä mitattiin Wireshark-ohjelmalla.

Tulosten perusteella saadaan kuva siitä, pienentääkö MQTT-protokolla siirrettävää tietomäärää. Jos tietomäärä on samaa kokoluokkaa molemmilla protokollilla, voi MQTT-protokollan käyttö olla mahdollista. 100 tavun hyötykuorma voi olla suuri MQTT-protokollan tapahtumapohjaisessa tiedonsiirrossa, mutta pieni kuorma käytettäessä S7-protokollaa. Koska S7-protokolla on valmistajakohtainen ja suljettu, sen kehysrakennetta ja sanoman käyttämää kokoa on helpoin arvioida siirtämällä tiedetty hyötykuorma ja analysoimalla verkkoliikennettä.

5.5.2 Suorituskyky S7-1200-sarjan logiikalla

Mitattaviksi resursseiksi valittiin protokollan muistin käyttö sekä ohjelman suoritusaika ohjelmoitavassa logiikassa. Jos kommunikoinnin toteutus käyttää liikaa logiikan muistia tai ohjelman suoritusaika kasvaa kohtuuttomasti, ei logiikkaan voida toteuttaa muuta suoritettavaa ohjelmaa.

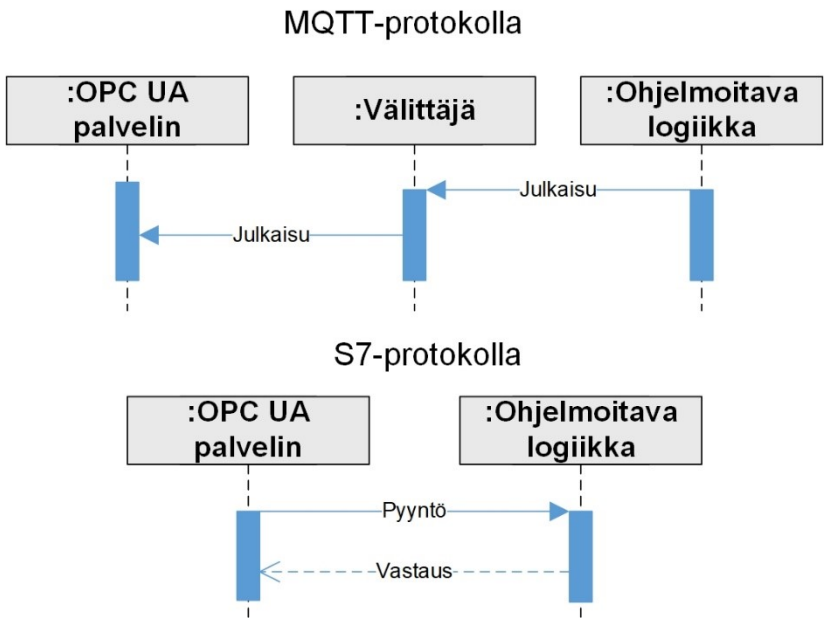
Resurssien käyttö mitattiin Siemensin kehitysympäristön työkaluilla. Kehitysympäristönä käytettiin TIA Portal V14 versiota. Muistin mittaaminen suoritettiin, kun protokollan tuki oli toteutettu kokonaisuudessaan ohjelmoitavaan logiikkaan. Suoritusaika mitattiin logiikan ollessa yhteydessä välittäjään ja suorittaessa toistuvasti sanomien julkaisua, jotta protokollan käsittelyn kuormittavuus ilmenee mittauksista. Julkaisujen palvelunlaadun taso oli 0 ja hyötykuorman koko pidettiin 100 tavussa.

5.5.3 Tiedonsiirron käsittelynopeuden mittaaminen

Tiedonsiirron käsittelyn nopeutta mitattiin MQTT- ja S7-protokollilla. Tietokoneeseen, jossa integraatiosovellus sekä OPC UA asiakassovellus sijaitsee, asennettiin NTP-palvelin (engl. Network Time Protocol). Ohjelmoitava logiikka määritettiin synkronoimaan kello NTP-palvelimen kanssa. Tiedonsiirtoon käytettävää aikaa mitattiin 10 ja 100 tavun hyötykuormilla. Sekä 10 että 100 tavun mittauksia suoritettiin 50 kertaa. Näistä mittauksista laskettiin 10 ja 100 tavun kerroille keskiarvot.

Tietokone, jossa oli asennettuna välittäjä sekä integraatiosovellus, ja ohjelmoitava logiikka olivat yhteydessä toisiinsa suoralla Ethernet-kaapelilla. Lähiverkossa ei ollut muita laitteita. Tällä tavoin pyrittiin minimoimaan viiveet sekä viiveiden erot. Testillä oli tarkoitus mitata tiedon käsittelyyn kuluvaa viivettä molemmilla protokollilla.

MQTT- ja S7-protokollat käyttävät eri kommunikointimalleja, joten samanlaista mittaustapaa ei käytetty. Mittaustavalla haluttiin mitata tiedonsiirron suorituksen nopeutta, ja sitä kuinka nopeasti tieto on käytettävissä pyynnön tai julkaisun suorituksesta. Mittausaikaan sisältyvät tiedonsiirrot esitetään kuvassa 29. S7-protokollaa käytettäessä, mitattavaan aikaan sisältyi pyyntöön ja vastauksen saamiseen sekä käsittelyyn kulunut aika. MQTT-protokollaa käytettäessä, mitattavaan aikaan sisältyi ohjelmoitavan logiikan protokollan käsittely, sanoman lähetys välittäjälle sekä aika, joka kului, kun välittäjä välitti sanoman OPC UA -palvelimelle.



Kuva 29. *Tiedonsiirrot, jotka sisältyivät tiedonsiirrossa kuluvaan aikaan mittauksissa. Alhaalla koe S7-protokollalla ja ylhäällä koe MQTT-protokollalla.*

Mittaukset suoritettiin MQTT-protokollalla, kun ohjelmoitava logiikka suoritti julkaisua, jossa hyötykuormaan asetettiin kuluvan vuorokauden ajankohta millisekunteina. Kun sanoma oli käytettävissä OPC UA -palvelimella, laskettiin OPC UA -palvelimen ja hyötykuormassa olevan aikaleiman erotus. Kun mittaukset suoritettiin S7-protokollalle, OPC UA -palvelin tallensi kuluvan vuorokauden ajankohdan millisekunteina, kun palvelin lähetti pyynnön ohjelmoitavalle logiikalle. Kun vastaussanoma oli käytettävissä OPC UA -palvelimella, laskettiin sen hetken ja pyynnön lähetyshetken erotus.

6. TULOKSET JA NIIDEN ARVIOINTI

Tässä luvussa esitetään testijärjestelmällä suoritettujen mittauksien tulokset. Aliluvussa 5.1.1 esitetyn vaatimuksen 8 täyttyminen valitulla protokollalla varmistettiin testijärjestelmässä suoritetuilla mittauksilla, jotka ovat kuvattu aliluvuissa 5.5.1 – 5.5.3.

Testattavaksi kommunikointiprotokollaksi valikoitui vertailussa MQTT-protokolla. Sen vahvuuksiksi osoittautui erityisesti kommunikointimalli, yksinkertaisuus sekä ominaisuudet. Vertailun tulokset esitettiin aliluvussa 5.2. Protokollan tuen toteutuksen avulla arvioidaan sen käytettävyyttä tiedonkeruujärjestelmässä etäasemien ohjelmoitavissa logiikoissa. Tässä luvussa esitetään mittauksien tulokset. Mittaukset suoritettiin testausjärjestelmässä käyttäen MQTT- ja S7-protokollia, koska haluttiin tietää protokollan suorituskyky verrattuna ohjelmoitavassa logiikassa natiiviin kommunikointiprotokollaan. Testausjärjestelmä kuvattiin aliluvussa 5.4.

6.1 Siirrettävä tietomäärä

Suuren hyötykuormakoon, 100 tavua, sisältävien sanomien kehyskoot mitattiin MQTT- ja S7-protokollilta käyttäen Wireshark-ohjelmistoa. MQTT- ja S7-protokollan sanomat siirretään käyttäen TCP-yhteyttä, mutta S7 protokollan sanoma koostuu useammasta kerroksesta. Taulukossa 3 esitetään kehyksien koot MQTT- ja S7-protokollilla.

Taulukko 3. *S7- ja MQTT-kommunikointiprotokollien kehyskoot siirrettäessä 100 tavun hyötykuormaa.*

S7		**MQTT v.3.1.1**	
Kehys	Koko (tavua)	Kehys	Koko (tavua)
Ethernet	14	Ethernet	14
IPv4	20	IPv4	20
TCP	20	TCP	20
ISO on TCP	4	MQTT	124
ISO 8073/X.224	3		
S7 Communication	122		
Yhteensä	**183**		**178**

Kommunikointiprotokollien ylempien kerrosten kehykset olivat samat, jolloin myös kehyskoot olivat saman suuruiset. MQTT-protokollan kommunikoinnissa käytettiin 20 merkin, jolloin otsikkokenttien kanssa koko MQTT-sanoman kooksi tulee 24 tavua. Aiheen

koko vaikuttaa MQTT-paketin kokoon, jolloin tämä tulee pitää aiheita luodessa mielessä. 20 merkin aiheeseen saa sisällytettyä tarkan kuvauksen siirrettävästä tiedosta ja toisaalta lyhyempikin aihe on riittävä. Samankokoisella hyötykuormalla pakettikoot ovat melkein saman suuruiset, S7-protokollalla 183 tavua ja MQTT-protokollalla 178 tavua. Koska MQTT-protokollan tiedonsiirto voidaan toteuttaa tapahtumapohjaisesti, voidaan siirtää esimerkiksi vain muuttunut tieto. Tällöin sanomien koko voi olla pieni – vain muutaman tavun.

6.2 Suorituskyky S7-1200-sarjan logiikalla

Toteutetun S7-1200 MQTT-asiakkaan suorituskykytiedot mitattiin TIA Portal -kehitysympäristön työkaluilla. Mittaukset suoritettiin, kun logiikka ei suorittanut MQTT-sanoman julkaisua, sekä silloin kun ohjelmoitava logiikka suoritti 100 tavun hyötykuorman julkaisua toistuvasti. Tulokset esitetään molemmille tapauksille taulukossa 4.

Taulukko 4. *Ohjelmoitavan logiikan ohjelman suoritusaika, kun julkaisua ei suoritettu ja toistuvassa suorituksessa.*

	Ei julkaisua	**Julkaisu**
Lyhin ohjelman kiertoaika	1 ms	1 ms
Mittaushetken ohjelman kiertoaika	2 ms	2 ms
Pisin ohjelman kiertoaika	3 ms	5 ms

Ilman aktiivista MQTT-kommunikointia lyhimmäksi suoritusajaksi mitattiin 1 ms ja pisimmäksi 3 ms ohjelmoitavan logiikan käynnistyksestä. Kun 100 tavun hyötykuormaa lähetettiin toistuvasti, pisimmäksi suoritusajaksi mitattiin 5 ms. TIA Portal -kehitysympäristö esittää suoritusajan ms tarkkuudella. Tuloksista voidaan päätellä, että MQTT-asiakkaan toteutus ohjelmoitavaan logiikkaan ei aiheuttanut kohtuutonta suoritusajan kasvua. Kuitenkin molempien kiertoaikojen pituus mittaushetkellä on ollut 2 ms, eli protokollan käsittely ei vaikuta juurikaan logiikan prosessorin kuormitukseen.

Vaikka tiedonsiirto ei vaikuttanut ohjelman suoritukseen merkittävästi, täytyy protokollan toteutuksen käytettävyyttä arvioida lisäksi sillä, kuinka se käyttää ohjelmoitavan logiikan muistia. Siemensin 1200-sarjan ohjelmoitavien logiikoiden muistit jaetaan 3 tyyppiin: kuorma-, työ- ja säilytettävään muistiin. Kuormamuisti on pysyväismuistia ja se on ohjelmaa, tietolohkoja ja laitekonfiguraatiota varten. Kuormamuisti on laajennettavissa muistikortilla. Työmuisti on lyhytkestoista muistia, joka sisältää suorituksenaikaisen ohjelman

ja tietolohkot. Työmuisti on integroitu prosessoriin, jolloin se ei ole laajennettavissa. Säilytettävä muisti on pysyväismuistia, jota käytetään rajallisen tietomäärän säilyttämiseen virtakatkoksien aikana. [89]

Muistin käyttö tarkastettiin protokollan tuen toteuttamisen jälkeen. Muistin käytön tulokset esitetään taulukossa 5. Taulukon arvojen yksikkö on tavu.

Taulukko 5. *Toteutuksen muistin käyttö mallin S7-1215C ohjelmoitavasta logiikasta. Muistimäärät tavuina.*

	Kuormamuisti	**Työmuisti**	**Säilytettävä muisti**
Käytössä	218 372	32 396	278
Käytettävissä	4 194 304	128 000	10 240
Suhteellinen käyttö	5,2 %	25,3 %	2,7 %

Toteutus käytti suhteellisesti eniten ohjelmoitavan logiikan työmuistia. Työmuisti on integroitu, eikä ole laajennettavissa. MQTT-protokollasta oli toteutettu sekä julkaisu, että tilaaminen. Toteutuksen käyttämistä mitta-asemissa voidaan pitää mahdollisena samanlaisella logiikkamallilla. Ohjelmaa voidaan muokata lisäksi siten, että tilaaminen poistetaan ohjelmasta kokonaan, jolloin muistia on käytettävissä enemmän. Jos toteutusta käytetään esimerkiksi pumppaamoissa, joihin lähetetään tietoa ja joilla toteutetaan prosessinohjaus, täytyy käyttää työmuistiltaan isompaa logiikkamallia. Käytännössä muulle ohjelmalle ei ole käytössä 74,7 % muistista, koska ohjelmamuutoksien takia ohjelmoituun logiikan muistiin täytyy jättää tilaa, tapauskohtaisesti esimerkiksi 10–20 %. Tällöin käytettäväksi jää 64,7–54,7 % työmuistista. Siksi logiikaksi pitää valita Siemensin S7-1200-sarjasta työmuistiltaan suurempi logiikkamalli tai Siemensin S7-1500-sarjan ohjelmoitavia logiikkoja. Hyvä puoli asiakasohjelmassa oli kuitenkin valmiiksi kattavat ominaisuudet, jolloin niiden lisäämiseen ei tarvitse varata logiikan muistia.

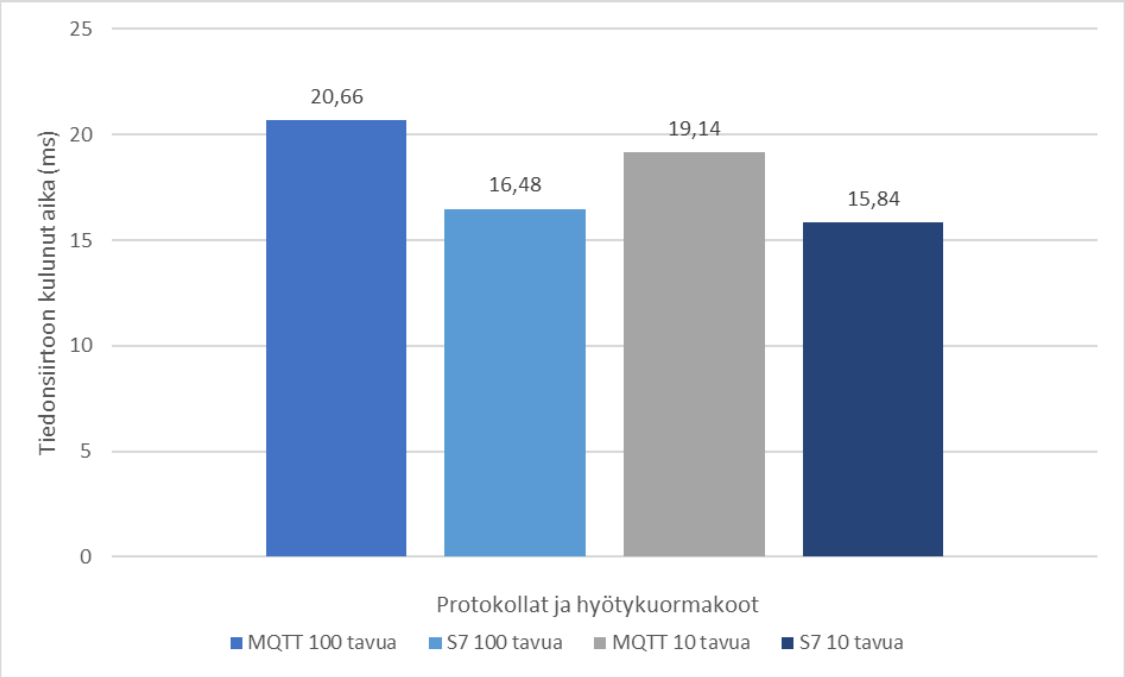
Saadun tuloksen perusteella voidaan todeta, että monimutkaisemman protokollan tuen toteuttaminen edulliseen ohjelmoitavaan logiikkaan ei ole toteutettavissa siten, että logiikkaa voidaan käyttää mittaustietojen julkaisemiseen sekä prosessin ohjaamiseen. Esimerkiksi OPC UA PubSub -protokollan toteuttaminen kokonaisuudessaan, käyttäessä MQTT-protokollaa, olisi haasteellista S7-1200 logiikan kaltaisilla edullisilla ohjelmoitavilla logiikoilla.

Testissä olleen ohjelmoitavan logiikan laiteohjelmiston version takia TLS-salauksen vaikutus ohjelmoitavan logiikan resurssien käyttöön jäi jatkotutkittavaksi. Salausta käytettäessä prosessori suorittaa lähetettävän tiedon salaamisen sekä vastaanottaessa salauksien purkamisen. Tällöin voidaan olettaa, että tällä on vaikutusta suorituskykyyn. Lisäksi

ohjelmoitavaan logiikkaan annetaan sertifikaatti ja tämä käyttää muistia. Ei haittaa, jos sertifikaatti käyttää kuormamuistia, mutta työmuistin käyttö ohjelmoitavan logiikan käytettävyyden kannalta on haitallista.

6.3 Tiedon käsittelynopeus

Toteutetulla integraatiolla ja ohjelmoitavan logiikan MQTT-asiakassovelluksella testattiin vielä tiedon käsittelyn nopeutta. Nopeus testattiin myös käyttäen integrointisovelluksessa S7-protokollaa, jotta tuloksille saatiin vertailuarvot. Testatut hyötykuormien koot olivat 10 ja 100 tavua. MQTT-kommunikoinnin palvelunlaadun taso oli 0. Mittauksia suoritettiin 50 kappaletta ja niistä laskettiin keskiarvo. Kuvassa 30 esitetään saadut tulokset MQTT- ja S7-protokollilla 10 ja 100 tavun hyötykuormilla.



Kuva 30. *Tiedonsiirtomittauksien tulokset MQTT- ja S7-protokollilla 10 ja 100 tavun hyötykuormilla.*

Siemensin ohjelmoitavat logiikkojen kommunikointiprosessorit kommunikoivat suoraan S7-protokollalla, jolloin sen käsittelyä ei tarvitse toteuttaa logiikkaohjelmaan. Muiden kuin integroitujen kommunikointiprotokollien käsittely on toteutettava erikseen logiikkaohjelmaan. Tästä syystä S7-protokolla saa Siemensin logiikoissa suurta etua verrattuna muihin protokolliin. Tämä on nähtävissä myös mittaustuloksista, sillä molemmilla hyötykuormilla S7-protokollalla tieto oli käytettävissä OPC UA -palvelimella nopeammin kuin MQTT-protokollalla. Tiedonsiirto ja käsittely oli nopeampaa, vaikka S7-protokolla käyttää

asiakas-palvelin-kommunikointimallia. Asiakas-palvelin-mallissa OPC UA -palvelin lähetti pyynnön ohjelmoitavalle logiikalle, ja logiikka vastasi pyyntöön lähettäen hyötykuorman. Pyyntöön kulunut aika sisältyi mitattavaan aikaan.

Tiedonsiirron ja käsittelyjen kestojen ero ei ole kuitenkaan merkittävä, eikä poissulje MQTT-protokollan käytettävyyttä ohjelmoitavissa logiikoissa. Todellisessa tilanteessa S7-protokollaa käytettäessä asiakasohjelmat joutuvat lähettämään pyynnöt ohjelmoitaville logiikoille tietyin väliajoin. Tiedonsiirto ei ole tapahtumapohjaista ja asiakasohjelma ei tiedä, milloin tieto on muuttunut. Tätä viivettä ei tullut mukaan näissä mittauksissa. Vaikka testauksessa S7-protokollan tiedonsiirto oli nopeampaa kuin MQTT-protokollalla, todennäköisesti hitaassa ja viiveellisessä verkossa, joissa viivettä on esimerkiksi joitain satoja millisekunteja, tiedonsiirron kesto kasvaa enemmän S7-protokollalla pyyntöjen lähettämisen takia. Samalla siirretään tietoa, joka ei ole muuttunut ja tietoa siirretään turhaan. Kun tiedonsiirto toteutetaan tapahtumapohjaisesti, siirretään vain tarpeellinen ja muuttunut tieto, jolloin vähennetään siirrettävää tietomäärää. Myös sanomakokojen pitäminen pieninä on mahdollista. Lisäksi, jos asiakkaita on useita, joudutaan tieto pyytämään kiertokyselynä ja viive kasvaa edelleen. MQTT-asiakkaat voivat lähettää tiedot samanaikaisesti.

MQTT-protokollalla eri palvelunlaatutasoilla kättelyiden määrä kommunikoivien osapuolten välillä vaihtelee. Testin palvelunlaatutasolla julkaiseva osapuoli ei varmista, että vastaanottaja vastaanottaa sanoman. Yksittäisen mittaustiedon kohdalla tämä on hyväksyttävää. Ainakin kerran -palvelunlaatutasolla vastaanottaja lähettää kuittaussanoman ja täsmälleen kerran -palvelunlaatutasolla suoritetaan 4-osainen kättely. Nämä pidentävät siirrettävän tiedon käsittelyaikaa erityisesti hitailla yhteyksillä, jos viiveet ovat esimerkiksi joitain satoja millisekunteja. Tuolloin yksittäisen julkaisun tapauksessa esimerkiksi 4-osaisessa kättelyssä käsittelyyn kuluva aika moninkertaistuu. Toisaalta asiakkaan tai välittäjän ei tarvitse jäädä odottamaan, että julkaisu käsitellään loppuun. Asiakas voi siirtyä tekemään uutta julkaisua kuittaussanomien välissä.

Testauksessa käytettiin ActiveMQ Artemis väliohjelmistoa, joka tukee useaa protokollaa sekä ohjelmointirajapintoja. Ohjelma on monipuolinen ja mahdollisesti turhan raskas oikeassa käyttöympäristössä, jos käytetään vain MQTT-protokollaa. Tuolloin kannattaa käyttää MQTT-protokollalle tarkoitettua välittäjää, esimerkiksi Eclipse Mosquittoa, joka on saatavilla lähteestä [91]. Kevyt välittäjä ei kuormaa paljoa prosessoria, jolloin tämä voi pienentää välitykseen kuluvaa aikaa.

6.4 Jatkotutkimus ja kehitystarpeet

Sovellus, jossa integroitiin MQTT-protokolla OPC UA -palvelimeen, osoittautui arkkitehtuurin puolesta hyväksi. Lisäksi käsittelijät, joilla voidaan hallita välitettäviä tietoja, osoittautui käyttökelpoiseksi ratkaisuksi. Käsittelijöiden luontiin on hyvä kehittää esimerkiksi konfigurointi, esimerkiksi XML-tiedostolla, joka tuotaisiin sovellukseen ohjelman alustuksessa. Vaihtoehtoisesti asetuksien asetteluun voi toteuttaa käyttöliittymän.

S7-1200 julkaisijan sanomienkäsittely oli toteutettu merkkijonoja käyttämällä. Tällöin määritellyt muuttujat käyttävät tarpeettomasti muistia. Siemensin logiikan merkkijonot käyttävät sen verran muistia, kuin merkkijonon alustuksessa määritellään, riippumatta käytetystä merkkimäärästä. Merkkijonot pitää muuttaa julkaisuista sen tietotyypin vaatimaan muistimäärään. Aiheet ovat merkkijonoja, mutta niiden koko kannattaa pitää mahdollisimman lyhyinä. MQTT-asiakas voidaan myös jatkokehittää tukemaan MQTT-protokollan versiota 5.0, jolloin voidaan käyttää aiheiden aliasta. Tuolloin 16-bittinen kokonaisluku saadaan vastaamaan tiettyä aihetta.

Lisäksi jatkotutkittavaksi jäi S7-1200-sarjan logiikkojen TLS-salauksen käytön vaikutus logiikan suorituskykyyn ja muistin käyttöön. Ominaisuus tuli vuonna 2019 julkaistuun laiteohjelmiston versioon 4.3 [92] ja testikäytössä olleen logiikan laiteohjelmiston versio oli 4.1. Tästä syystä suorituskykytestiä ei voitu suorittaa salatulla kommunikoinnilla.

Valituksi tulleella MQTT-protokollalla aikaleima tai mittaus- sekä tilatiedon laatu pitää sisällyttää hyötykuormaan. Laatutieto ja aikaleima voidaan esittää siten, että ne käyttävät aina tietyn verran tilaa. Tästä syystä nämä tiedot voidaan laittaa hyötykuorman alkuun, sillä mittaus- tai tilatietojen vaatima tila vaihtelee tietotyypin mukaan. Testijärjestelmän aiheiden määrittelytapa, jossa OPC UA -muuttuja vastaa tiettyä aihetta, on yksinkertainen ja helppo toteuttaa. Tällöin aihe vastaa erillistä tilatietoa tai mittausta.

On kuitenkin huomioitava, että automaatiojärjestelmässä voi tapahtua useita samanaikaisia tapahtumia. Testaussovelluksen toteutus ei ota tätä huomioon. Jos jokainen aihe vastaa esimerkiksi tiettyä mittausta, täytyy etäaseman ohjelmaan toteuttaa jonotus tai muisti, jotta kaikki tapahtumat varmasti julkaistaan. Tällä tavalla toteutettuna reaaliaikaisuus kuitenkin kärsii. Lisäksi tapahtuman ajanhetki on otettava erikseen muistiin ja asetettava julkaisuun, kun tapahtuma saa julkaisuvuoron. Tällä estetään väärän ajanhetken antaminen tiedonkeruujärjestelmään. Väärä aikaleima aiheuttaa vääriä johtopäätöksiä tapahtumien kulusta.

Vaihtoehtoisena ratkaisuehdotuksena tälle voidaan hakea luvussa 3 esitellystä OPC UA:n PubSub-määrittelystä. Siinä julkaistavaan sanomaan kootaan joukko tietopaket-

teja, jotka sisältävät usean mittauksen arvon, laatutiedon ja aikaleiman. Eri tiedot on kuitenkin tunnistettava sanomasta. Tähän PubSub-määrittely, MQTT-protokollan tapauksessa, määrittelee käyttämään JSON-muotoista sanoman esitystapaa tai OPC UA Binary -koodattua sanomaa.

JSON-muotoinen sanoma olisi helposti ymmärrettävissä. Mittaukset voidaan yksilöidä sanomaan esimerkiksi position perusteella. Tässä toteutustavassa julkaisun aihe ei enää yksilöisi tapahtumaa, vaan aiheella yksilöitäisiin julkaisija. Tekstipohjainen sanoma kasvattaa kuitenkin sanomien kokoa, mutta suurempi ongelma on sanomien koostamisen lisämuistin tarve. Jo pelkästään protokollan tuen toteutus käyttää 25,3 % työmuistista. Tästä syystä pelkästään kommunikointiin ei ole järkevää käyttää paljoa lisämuistia, joten PubSub-protokollaa kannattaa hyödyntää vain osittain.

OPC UA Binary -koodatut sanomat olisivat mahdollisesti pienempiä kuin JSON-muotoiset sanomat. OPC UA Binary -koodattujen sanomien rakenteen ohjelmointi, kokonaisuudessaan määrittelyn mukaisesti ohjelmoitavaan logiikkaan, käyttäisi työmuistia. Tässäkin tapauksessa määrittelyä kannattaa hyödyntää osittain, ja sanomaan sisällyttää vain ne tiedot, joita tarvitaan, kuten aikaleima tai laatutiedot.

OPC UA PubSub -määrittelyjen mukaisesti saataisiin myös toteutettua tietoturva ja päästä-päähän eheys, jos käytetään jaettuja avaimia ja allekirjoitusta. Turva-avainpalvelun ja määrittelyn mukaisen salauksen toteuttaminen logiikkaan vaatisi resursseja. Lisäksi turva-avainpalvelun avaimien jako vaatii suojatun yhteyden, joten näiden toteuttaminen ohjelmoitavaan logiikkaan vaatisi resursseja.

7. YHTEENVETO

Tämän työn tarkoituksena oli tutkia IoT-järjestelmissä käytössä olevia sovellustason protokollia ja valita kaukolämpöjärjestelmien automaatioon tarkoitetun tiedonkeruujärjestelmän käyttöön soveltuvin protokolla. Tutkimuksen kohteena olivat seuraavat protokollat: AMQP, CoAP, MQTT ja HTTP. Käytettävän protokollan valintaan vaikuttivat järjestelmän tiedonsiirtoon asetetut vaatimukset. Asetetut vaatimukset tulivat tavanomaisista ja hajautettujen automaatiojärjestelmien tietoverkoille asetetuista vaatimuksista, sekä IIoT-järjestelmille asetetuista tietoverkkojen vaatimuksista.

Protokollan valinnan jälkeen toteutettiin prototyyppi, jossa integroitiin testattavan protokollan asiakaskomponentti OPC UA -palvelimeen. Lisäksi kehitettiin internetistä ladattavissa olevaa logiikan protokollan asiakasohjelmaa. Integraatiosta ja asiakasohjelmasta muodostettiin testausjärjestelmä, jolla oli tarkoitus testata protokollan käytettävyyttä ohjelmoitavissa logiikoissa sekä OPC UA:n integraatiossa. Testausympäristö antoi kokemusta arkkitehtuurista. Kokeet antoivat hyvän kuvan protokollan käytettävyydestä integraatiossa OPC UA:n kanssa ja ohjelmoitavissa logiikoissa. Tässä luvussa käydään läpi tutkimuskysymykset ja tutkimuksen tuottamat vastaukset.

Työn suurimmat haasteet olivat tutkimuksen kohteena olevien protokollien ominaisuuksien arviointi määrittelyjen ja aiheesta löytyvien aiempien tutkimuksien perusteella. Lähteissä saattoi olla puutteellisesti kerrottu, mitä protokollan versiota tutkija on tutkimuksessaan käyttänyt. Tällä oli suuresti merkitystä esimerkiksi AMQP-protokollan tapauksessa, koska AMQP-protokollan eri versiot poikkeavat suuresti toisistaan. Toisaalta suurta helpotusta työhön toi se, että valituksi tulleelle MQTT-protokollalle löytyy paljon valmiita asiakassovelluksia ja välitysohjelmistoja, jolloin näitä ei tarvinnut kokonaisuudessaan toteuttaa itse.

1. *Mikä kommunikointiprotokolla täyttää järjestelmälle asetetut vaatimukset parhaiten tutkimuksen kohteena olevista protokollista?*

Yksikään tutkimuksen kohteena olevista protokollista eivät täyttäneet kaikkia vaatimuksia. Tutkimuksen kohteena olevista protokollista testausympäristöön toteutettavaksi valikoitui MQTT-protokolla. MQTT-protokollan vahvuuksiksi osoittautui sen käyttämä julkaise-tilaa-kommunikointimalli, yksinkertaisuus ja ominaisuudet. AMQP-protokollalla on toteutettavissa sama kommunikointimalli ja luotettavuustasot, mutta monimutkaisemman rakenteen vuoksi, otsikkokentän koko kasvaa moninkertaiseksi verrattuna MQTT-

protokollaan. Monimutkainen rakenne antaa viitteitä myös siitä, että se ei olisi toteutettavissa järkevän resurssienkäytön puitteissa edulliseen ohjelmoitavaan logiikkaan. CoAP-protokollan etuna on yksinkertaisuus sekä keveys, mutta siinä on pieniä puutteita esimerkiksi luotettavuudessa. HTTP-protokollan kehyskokoa määrittely ei kerro, mutta todennäköisesti otsikkokentästä kasvaisi useamman tavun mittainen. Lisäksi HTTP/1.1 on tekstipohjainen, jolloin sanomat voivat kasvaa suuriksi. Kommunikointimallina on myös asiakas-palvelin, mikä voi aiheuttaa haasteita järjestelmän skaalautuvuuteen. Vaikka MQTT-protokolla täytti vaatimukset parhaiten, sen sanomassa ei kuljeteta muun muassa aikaleimaa. Tätä varten esitettiin jatkokehitysehdotuksia aliluvussa 6.4.

2. *Onko protokollan tuki toteutettavissa ja minkälainen on protokollan käytettävyys S7-1200-sarjan ohjelmoitavassa logiikassa resurssien käytön näkökulmasta?*

Jotta voidaan varmistua, että kommunikointiprotokollan käytöstä olisi hyötyä sovelluksessa täytyy todentaa protokollan suorituskyky järjestelmässä. Testausjärjestelmään valittiin S7-1200-sarjan ohjelmoitava logiikka, koska se olisi myös hyvä vaihtoehto kehitettävään tiedonkeräysjärjestelmään.

MQTT-protokolla on yleinen IIoT-käytössä ja tästä syystä sen käyttämiseen löytyy valmiita ohjelmistokomponentteja useille alustoille. Myös Siemensin ohjelmoitaville logiikoille löytyy valmiita MQTT-asiakasohjelmia. Tässä tutkimuksessa hyödynnettiin valmista lähdekoodia, jossa oli toteutettu MQTT-julkaisija. Jotta tietoa voidaan siirtää kahteen suuntaan, ohjelmaan toteutettiin lisäksi aiheiden tilaaminen.

Kun MQTT-protokollan tuki oli toteutettu ohjelmoitavaan logiikkaan, mitattiin sen resurssien käyttö ohjelmoitavassa logiikassa. Ohjelman kuormamuistin käyttö ei aiheuta ongelmia jatkossa, sillä ohjelma käytti sitä vain 5,2 %. Ohjelma käytti työmuistia 25,3 %, jolloin prosessinohjauksen toteuttaminen samaan logiikkaan on haasteellista ja suuremman saman S7-1200-sarjan logiikan tai S7-1500-sarjan logiikan käyttäminen on tarpeen. Pelkkien mittaus- tai tilatietojen julkaisu käytetyllä työmuistimäärällä on mahdollista. Lisäksi MQTT-asiakasta voidaan jatkokehittää siten, että se käyttää muistia vähemmän.

Ohjelman suoritusaikoihin protokollan toteutus ei aiheuttanut juurikaan nousua, ja siltä osin käyttö ohjelmoitavassa logiikassa on mahdollista. Koska S7-protokolla on Siemensin logiikoissa valmiiksi tuettuna, sen käyttäminen ei lisää muistin käyttöä, koska tälle ei tarvitse toteuttaa käsittelyä erikseen. S7-protokollaa ei kuitenkaan tue suoraan muiden valmistajien logiikat ja tarvitaan avoin protokolla kommunikointiin myös muiden valmistajien laitteille.

3. *Minkälainen on valitun protokollan suorituskyky S7-1200-sarjan ohjelmoitavassa logiikassa ja minkälainen sen suorituskyky on verrattuna S7-protokollaan?*

Lisäksi protokollilta mitattiin tiedonsiirron nopeutta. Testit suoritettiin siten, että 10 ja 100 tavun hyötykuormia julkaistiin 50 kertaa ja mitattiin, kuinka nopeasti tieto oli käytettävissä OPC UA -palvelimella. Saadut tulokset ovat mittauskertojen keskiarvoja. Sekä 10 ja 100-tavun kuormalla MQTT-protokolla suoriutui tiedonsiirrosta hitaammin. Koska MQTT-protokollan käsittely on toteutettu suoritettavaan logiikkaohjelmaan, sanomien käsittelyyn kuluva aika näkyy kommunikointiin kuluvassa ajassa. S7-protokollaa käytettäessä Siemensin ohjelmoitavissa logiikoissa tätä viivettä ei tule, koska logiikkaohjelmaan ei tarvitse erikseen toteuttaa protokollan käsittelyä. MQTT:tä käytettäessä järjestelmässä on lisäksi välittäjä, joka välittää sanoman OPC UA -palvelimelle. Kommunikointiin kuluvat ajat eivät silti ole suuret. Jos käytössä olisi hidas tietoliikenneyhteys, jonka viive olisi satoja millisekunteja, kääntyisi kommunikoinnin nopeus MQTT-protokollan eduksi. Tuolloin käsittelyyn kuluva aika ei kasva, mutta tiedon siirtämiseen kuluva aika tiedonsiirtoverkoissa kasvaa. S7-protokollalla tiedonsiirtoon kuluu aikaa pyyntöihin ja vastauksiin. MQTT-protokollalla tiedonsiirto on pääpiirteittäin yksisuuntaista.

Kokemusta saatiin lisäksi käytettävästä järjestelmäarkkitehtuurista. MQTT-asiakkaan integrointi OPC UA -palvelimen kanssa osoittautui yksinkertaiseksi toteuttaa. Tällä tavoin ei tarvitse toteuttaa OPC UA -palvelupohjaista arkkitehtuuria, eikä tarvitse erikseen toteuttaa OPC UA:n lukuisia palveluita. Jatkotutkittavaa ja -kehitettävää täytyy tehdä muun muassa palvelimen konfiguraation ja käsittelijöiden lisäämiseksi.

Jatkokehitystarpeita ilmeni myös julkaisujen muodostamiseen. Ehdotus toteutustavaksi kehitettävässä järjestelmässä olisi sama kuin testijärjestelmässä, jossa julkaisun aihe vastaisi OPC UA -muuttujaa. Julkaisut suoritetaan mittauksen kynnysarvon ylittävästä muutoksesta sekä tilatiedon muuttumisesta. Automaatiojärjestelmässä voi kuitenkin tulla samanaikaisia tapahtumia. Edellä mainitulla toteutustavalla logiikkaohjelmaan täytyy toteuttaa jonotus, jotta kaikki julkaisut tehdään. Sanomien alkuun on yksinkertaista sisällyttää aikaleima ja laatutieto.

Vaihtoehtoisena toteutustapana voidaan pitää OPC UA:n määrittelyn PubSub-osan hyödyntämistä. Osassa määritellään, kuinka MQTT-sanomaan sisällytetään OPC UA-palvelimen julkaisu. MQTT hyötykuorman rakentamiseen voidaan käyttää yksinkertaista JSON-mallia, tai OPC UA Binary -koodausta, johon yhteen sanomaan sisällytetään lähetyshetken aikaleima, järjestysnumero sekä useita mittaus- tai tilatietoja laatutietoineen. Kokonaisuudessaan PubSub-määrittelyn toteuttamista ei pidetä järkevänä, sillä pelkän MQTT-protokollan tuen toteuttaminen käytti integroitua työmuistia 25,3 %, joten lisämuistin käyttöä ei pidetä suositeltavana.

LÄHTEET

[1] N. Buecker, A 5-step approach to IIoT success, 20.9.2016, Saatavissa (viitattu 12.11.2019): https://www.plantengineering.com/articles/a-5-step-approach-to-iiot-success/

[2] H. Jan, D. Zhe Lou, C. Whitehead, S. Germanos, W. Qiu, M. Hilgner, Industrial networking enabling IIoT communication, Industrial Internet Consortium, 30.8.2018, Saatavissa: https://www.iiconsortium.org/pdf/Industrial_Networking_Enabling_IIoT_Communication_2018_08_29.pdf.

[3] Q. Yang, J. A. Barria, T. C. Green, Communication Infrastructures for Distributed Control of Power Distribution Networks, IEEE Transactions on Industrial Informatics, 2011, Vol. 7, Iss.2, pp. 316-327.

[4] B. Galloway, G.P. Hancke, Introduction to Industrial Control Networks, IEEE Communications Surveys & Tutorials, 2013, Vol. 15, Iss.2, pp. 860-880.

[5] N. Naik, Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP, IEEE, 2017, Saatavissa: https://ieeexplore.ieee.org/document/8088251

[6] M. Iglesias-Urkia, A. Orive, M. Barcelo, A. Moran, J. Bilbao, A. Urbieta, Towards a lightweight protocol for Industry 4.0: An implementation based benchmark, 2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM), 2017, pp. 1-6.

[7] Y. Chen, T. Kunz, Performance evaluation of IoT protocols under a constrained wireless access network, IEEE, 2016, Saatavissa: https://ieeexplore.ieee.org/document/7496622

[8] Examples for the SIMATIC S7-1200/S7-1500 Web Server, Siemens, 2018, Saatavissa (viitattu 5.5.2019): https://cache.industry.siemens.com/dl/files/496/68011496/att_959527/v2/68011496_Examples_for_S7Web-Server_DOC_v21_en.pdf

[9] A. Nipper, MQTT's role as an IoT message transport, Control Engineering, 7.1. 2019, Vol. 66, Iss.1, pp. 20.

[10] D. Bruckner, M. Stanica, R. Blair, S. Schriegel, S. Kehrer, M. Seewald, T. Sauter, An Introduction to OPC UA TSN for Industrial Communication Systems, Proc IEEE, 2019, Vol. 107, Iss.6, pp. 1121-1131.

[11] M. Åkerman, Implementing shop floor IT for industry 4.0, Chalmers University of Technology, 2018, Saatavissa: https://www.researchgate.net/publication/326224890_Implementing_Shop_Floor_IT_for_Industry_40/link/5b3f2650aca27207851c6c70/download.

[12] K. Ashton, That 'Internet of Things' Thing, 22.6.2009, Saatavissa (viitattu 18.4.2019): https://www.rfidjournal.com/articles/view?4986

[13] Overview of the internet of things, The International Telecommunication Union, 2012, Saatavissa: https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=y.2060.

[14] What is IIoT? The Industrial Internet of Things, 13.7.2018, Saatavissa (viitattu 18.4.2019): https://inductiveautomation.com/resources/article/what-is-iiot

[15] S. Weallans, IIoT And Industry 4.0: The Basics You Need to Know, 4.9.2018, Saatavissa (viitattu 18.4.2019): https://www.sensorsmag.com/components/iiot-and-industry-4-0-basics-you-need-to-know

[16] A. Nasrallah, A.S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, H. ElBakoury, Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research, IEEE Communications Surveys & Tutorials, 2019, Vol. 21, Iss.1, pp. 88-145.

[17] S.K. Rao, R. Prasad, Impact of 5G Technologies on Industry 4.0, Wireless Personal Communications, 2018, Vol. 100, Iss.1, pp. 145-159.

[18] D.E. Comer, Computer Networks and Internets, Pearson Education Inc., New Jersey, USA, Fifth ed., 2009, 637 p.

[19] B. Adryan, T. Konigseder, P. Fremantle, The Technical Foundations of IoT, Artech House, Norwood Massachusetts, USA, 2017, 494 p.

[20] A. Semle, IIoT Protocols to Watch, 26.10.2015, Saatavissa (viitattu 18.4.2019): https://www.automation.com/library/white-papers/iiot-protocols-to-watch

[21] J. Rajive, P. Didier, J. Jimenez, T. Carey, The industrial internet of things volume G5: Connectivity framework, Industrial Internet Consortium, 2018, Saatavissa: https://www.iiconsortium.org/pdf/IIC_PUB_G5_V1.01_PB_20180228.pdf.

[22] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, A. Hahn, Guide to industrial control systems (ICS) security, U.S. Dept. of Commerce, National Institute of Standards and Technology, 2007, Saatavissa: http://purl.access.gpo.gov/GPO/LPS97148.

[23] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications, COMST, 2015, Vol. 17, Iss.4, pp. 2347-2376.

[24] Five key elements for effective IIoT implementation, industrial ethernet book, 2018, Vol. 2, Iss.104, pp. 14-15.

[25] C.C. Byers, Architectural Imperatives for Fog Computing: Use Cases, Requirements, and Architectural Techniques for Fog-Enabled IoT Networks, IEEE Communications Magazine, 2017, Vol. 55, Iss.8, pp. 14-20.

[26] Skkynet, IIoT Protocol Comparison, Saatavissa (viitattu 21.3.2019): https://skkynet.com/iiot-protocol-comparison/

[27] P. Beecher, How wireless mesh networks will drive widespread IoT growth, industrial ethernet book, 5. 2018, Iss.106, pp. 50.

[28] E.D. Knapp, J.T. Langill, Industrial Network Security, 2nd Edition, Syngress, 2014, 460 p.

[29] S. Singh, N. Singh, Internet of Things (IoT): Security challenges, business opportunities & reference architecture for E-commerce, IEEE, 2015, Saatavissa: https://ieeexplore.ieee.org/document/7380718

[30] M.A.K. Khan, K. Salah, IoT security: Review, blockchain solutions, and open challenges, Future generation computer systems, 26.11. 2017, Vol. 82, pp. 395-411.

[31] Q. Mahmoud, Middleware for Communications, John Wiley & Sons, 29.6.2004, 487 p.

[32] R. Monson-Haefel, D. Chappell, M. Richards, Java Message Service, O'Reilly Media, 2nd ed., 12.6.2009, 330 p.

[33] I. Gorton, Essential Software Architecture, Springer, 2nd ed., 5.5.2011, 239 p.

[34] S. Tarkoma, Publish/subscribe systems : design and principles, John Wiley & Sons Ltd, 2012, 341 p.

[35] S.C. Yadav, S.K. Singh, An introduction to client/server computing, New Age International, New Delhi, 2009, 200 p.

[36] Unified Architecture - OPC Foundation, OPC Foundation, Saatavissa (viitattu 21.3.2019): https://opcfoundation.org/about/opc-technologies/opc-ua/

[37] W. Mahnke, S. Leitner, M. Damm, OPC Unified Architecture, Springer, Berlin Heidelberg, 2009, 351 p.

[38] M. Damm, OPC UA technical update: PubSub model and time sensitive networks (TSN), OPC Foundation, 13.11.2018, Saatavissa: https://www.automaati-oseura.fi/sas/jaostot/opc/tapahtumat/opc-day-finland-2018/.

[39] OPC unified architecture part 1 :Overview and concepts, OPC Foundation, 22.11.2017, Saatavissa: https://opcfoundation.org/developer-tools/specifications-unified-architecture.

[40] P. Drahos, E. Kucera, O. Haffner, I. Klimo, Trends in industrial communication and OPC UA, 2018, Saatavissa: https://www.researchgate.net/publication/324548588_Trends_in_industrial_communication_and_OPC_UA

[41] SIMATIC S7-1500 CPUs, Saatavissa (viitattu 13.10.2019): https://new.siemens.com/global/en/products/automation/systems/industrial/plc/simatic-s7-1500/cpus.html

[42] NJ/NX-series CPU unit OPC UA user's manual, Omron Corporation, 2018, Saatavissa: https://assets.omron.eu/downloads/manual/en/v5/w588_nx_nj-series_opc_ua_cpu_units_users_manual_en.pdf.

[43] OPC unified architecture part 3 :Address space model, OPC Foundation, 22.12.2017, Saatavissa: https://opcfoundation.org/developer-tools/specifications-unified-architecture.

[44] Embedded OPC UA Stack, Unified Automation, Saatavissa (viitattu 21.3.2019): https://documentation.unified-automation.com/uasdkhp/1.0.0/html/_l2_ua_address_space_concepts.html

[45] OPC unified architecture part 4: Services, OPC Foundation, 22.11.2017, Saatavissa: https://opcfoundation.org/developer-tools/specifications-unified-architecture/.

[46] OPC unified architecture part 6 :Mappings, OPC Foundation, 22.11.2017, Saatavissa: https://opcfoundation.org/developer-tools/specifications-unified-architecture.

[47] OPC unified architecture part 14 :PubSub, OPC Foundation, 6.2.2018, Saatavissa: https://opcfoundation.org/developer-tools/specifications-unified-architecture.

[48] OPC UA Roadmap - OPC Foundation, Saatavissa (viitattu 7.10.2019): https://opcfoundation.org/about/opc-technologies/opc-ua/opcua-roadmap/

[49] AMQP is the Internet Protocol for Business Messaging, OASIS, Saatavissa (viitattu 28.3.2019): https://www.amqp.org/about/what

[50] Business FAQs : AMQP, OASIS, Saatavissa (viitattu 8.4.2019): https://www.amqp.org/resources/business-faqs

[51] Standards OASIS, Saatavissa (viitattu 20.5.2019): https://www.oasis-open.org/standards#amqpv1.0

[52] OASIS advanced message queuing protocol (AMQP) version 1.0, OASIS Standard, 29.10.2012, Saatavissa: http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf.

[53] AMQP advanced message queuing protocol, version 0-9-1, AMQP, 13.11.2008, Saatavissa: http://www.amqp.org/specification/0-9-1/amqp-org-download.

[54] AMQP 0-9-1 Protocol Specification, Pivotal, Saatavissa (viitattu 15.4.2019): https://www.rabbitmq.com/protocol.html

[55] E. Rescorla, The transport layer security (TLS) protocol version 1.3, IETF, 2018, Saatavissa: https://tools.ietf.org/html/rfc8446.

[56] A. Melnikov, K. Zeilenga, Simple authentication and security layer (SASL), IETF, 2006, Saatavissa: https://tools.ietf.org/html/rfc4422.

[57] I. Grigorik, HTTP protocols, O'Reilly Media, Inc, 1st ed., 2017, 20 p.

[58] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, RFC 2616 - hypertext transfer protocol, IETF, 1999, Saatavissa: https://tools.ietf.org/html/rfc2616.

[59] M. Belshe, R. Peon, M. Thomson, Hypertext transfer protocol version 2 (HTTP/2), IETF, 5.2015, Saatavissa: https://tools.ietf.org/html/rfc7540.

[60] J. Franks, P.M. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, RFC 2617 - HTTP authentication: Basic and digest access authentication, IETF, 1999, Saatavissa: https://tools.ietf.org/html/rfc2617#section-3.1.

[61] B. Pollard, HTTP/2 in Action, Manning Publications, 1st ed., 2019, 416 p.

[62] Z. Shelby, K. Hartke, C. Bormann, The Constrained Application Protocol (CoAP), 2014, Saatavissa (viitattu 4.4.2019): https://tools.ietf.org/html/rfc7252

[63] D. Reynders, E. Wright, Practical TCP/IP and Ethernet networking, Elsevier Science, Boston, 2003, 306 p.

[64] C. Bormann, A.P. Castellani, Z. Shelby, CoAP: An Application Protocol for Billions of Tiny Internet Nodes, MIC, 2012, Vol. 16, Iss.2, pp. 62-67.

[65] M. Becker, K. Li, K. Kulanidithi, T. Pötsch, Transport of CoAP over SMS, 8.8.2014, Saatavissa: https://tools.ietf.org/id/draft-becker-core-coap-sms-gprs-05.html.

[66] C. Bormann, S. Lemay, H. Tschofenig, K. Hartke, B. Silverajan, B. Raymor, CoAP (constrained application protocol) over TCP, TLS, and WebSockets, 2018, Saatavissa: https://tools.ietf.org/html/rfc8323.

[67] W. Yanq, A WebRTC data channel transport for the constrained application protocol (CoAP), 19.4.2017, Saatavissa: https://tools.ietf.org/html/draft-groves-coap-webrtcdc-02.

[68] E. Rescorla, N. Modadugu, RFC - datagram transport layer security version 1.2, IETF, 2012, Saatavissa: https://tools.ietf.org/pdf/rfc6347.pdf.

[69] OASIS Standards, OASIS, Saatavissa (viitattu 29.3.2019): https://www.oasis-open.org/standards#mqtt-v5.0-os

[70] A. Banks, R. Gupta, MQTT version 3.1.1, OASIS Standard, 10.4.2014, Saatavissa: https://www.oasis-open.org/standards#mqttv3.1.1.

[71] MQTT Essentials: Part 1 - Introducing MQTT, HiveMQ, 12.1.2015, Saatavissa (viitattu 30.3.2019): https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/

[72] A. Banks, R. Gupta, E. Briggs, K. Borgendale, MQTT version 5.0, OASIS Standards, 7.3.2019, Saatavissa: https://www.oasis-open.org/standards#mqtt-v5.0-os.

[73] D. Soni, A. Makwana, A survey on mqtt: A protocol of internet of things(iot), 2017, Saatavissa: https://www.researchgate.net/publication/316018571.

[74] MQTT Security Fundamentals: Authentication with Username and Password, HiveMQ, 20.4.2015, Saatavissa (viitattu 31.3.2019): https://www.hivemq.com/blog/mqtt-security-fundamentals-authentication-username-password/

[75] MQTT and the NIST cybersecurity framework version 1.0, OASIS, 28.5.2014, Saatavissa: http://docs.oasis-open.org/mqtt/mqtt-nist-cybersecurity/v1.0/cn01/mqtt-nist-cybersecurity-v1.0-cn01.pdf.

[76] MQTT security, IBM, 28.2.2018, Saatavissa (viitattu 31.3.2019): https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.5.0/com.ibm.mm.tc.doc/tc00150_.htm

[77] MQTT Security Best Practices, Cirrus Link, 5.3.2018, Saatavissa (viitattu 22.3.2019): https://www.cirrus-link.com/mqtt-security-best-practices/

[78] MQTT Essentials Part 6: Quality of Service 0, 1 and 2, HiveMQ, 16.2.2015, Saatavissa (viitattu 31.3.2019): https://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels/

[79] MQTT Essentials: Part 7 - Persistent Sessions and Queuing Messages, 23.2.2015, Saatavissa (viitattu 31.7.2019): https://www.hivemq.com/blog/mqtt-essentials-part-7-persistent-session-queuing-messages/

[80] MQTT Essentials: Part 8 - Retained Messages, 2.3.2015, Saatavissa (viitattu 31.7.2019): https://www.hivemq.com/blog/mqtt-essentials-part-8-retained-messages/

[81] MQTT Essentials: Part 9 - Last Will and Testament, 9.3.2015, Saatavissa (viitattu 31.7.2019): https://www.hivemq.com/blog/mqtt-essentials-part-9-last-will-and-testament/

[82] S7comm, 13.5.2016, Saatavissa (viitattu 22.8.2019): https://wiki.wireshark.org/S7comm

[83] Node-OPCUA, Saatavissa (viitattu 30.5.2019): https://node-opcua.github.io/

[84] The MIT License, Open Source Initiative, 23.8.2019, Saatavissa (viitattu 23.8.2019): https://opensource.org/licenses/mit-license.html

[85] MQTT.js, Saatavissa (viitattu 30.5.2019): https://www.npmjs.com/package/mqtt

[86] Node.JS library for communication to Siemens S7 PLCs, Saatavissa (viitattu 2.7.2019): https://github.com/plcpeople/nodeS7

[87] Apache ActiveMQ artemis user manual, ActiveMQ, , Saatavissa: https://activemq.apache.org/components/artemis/documentation/.

[88] ActiveMQ - Running an ActiveMQ Broker, Saatavissa (viitattu 3.6.2019): https://activemq.apache.org/run-broker

[89] S7-1200 programmable controller, system manual, Siemens AG, 2016, Saatavissa: https://cache.industry.siemens.com/dl/files/465/36932465/att_106119/v1/s71200_system_manual_en-US_en-US.pdf.

[90] MQTT Publisher for SIMATIC CPU, 16.8.2018, Saatavissa (viitattu 25.6.2019): https://support.industry.siemens.com/cs/document/109748872/mqtt-publisher-for-simatic-cpu?dti=0&lc=en-WW

[91] Eclipse Mosquitto, Saatavissa (viitattu 25.7.2019): https://mosquitto.org/

[92] S7-1200 programmable controller - system manual 4.3.0, Siemens, 2019, Saatavissa: https://support.industry.siemens.com/cs/document/109764129/simatic-s7-s7-1200-programmable-controller?dti=0&lc=en-WW.

LIITE A: OPC UA TIETOTYYPIT

Taulukko 1. *OPC UA:n tietotyypit. Muokattu lähteestä [46].*

ID	Nimi	Kuvaus
1	Boolean	Kaksitilainen looginen arvo (tosi tai epätosi)
2	SByte	Kokonaisluku, joka voi saada arvoja väliltä -128 ja 127
3	Byte	Kokonaisluku, joka voi saada arvoja väliltä 0 ja 255
4	Int16	Kokonaisluku, joka voi saada arvoja väliltä -32 768 ja 32
5	UInt16	Kokonaisluku, joka voi saada arvoja väliltä 0 ja 65 535
6	Int32	Kokonaisluku, joka voi saada arvoja väliltä -2 147 483 648 ja 2 147 483 647
7	UInt32	Kokonaisluku, joka voi saada arvot väliltä 0 ja 4 294 967
8	Int64	Kokonaisluku, joka voi saada arvot väliltä -9 223 372 036 854 775 808 ja 9 223 372 036 854 775 807
9	UInt64	Kokonaisluku, joka voi saada arvot väliltä 0 ja 18 446 744 073 709 551 615
10	Float	IEEE-standardin mukainen perustarkkuuden (32 bittiä) liu-
11	Double	IEEE-standardin mukainen kaksoistarkkuuden (64 bittiä)
12	String	Unicode merkistön mukainen merkkijono
13	DateTime	Päivämäärä ja aika
14	Guid	16-tavuinen arvo globaaliin yksilöintiin
15	ByteString	Oktettijono
16	XmlElement	XML-elementti
17	NodeId	Solmutunniste OPC UA palvelimen osoiteavaruuteen
18	ExpandedNodeId	Solmutunniste, joka sallii URIn (Uniform Resource Identifier) spesifioimisen indeksin käytön sijaan
19	StatusCode	Numeerinen tunniste arvojen ja operaatioiden virheille ja tiloille
20	QualifiedName	Nimiavaruuden hyväksymä nimi
21	LocalizedText	Luettava teksti, jossa on vaihtoehtoinen paikkatunniste
22	ExtensionObject	Struktuuri, joka sisältää sovelluskohtaisen tiedotyypin. Tietotyyppi ei ole välttämättä vastaanottajan tunnistettavissa.
23	DataValue	Tieto, joka sisältää siihen liittyvät tilatiedot ja aikaleimat
24	Variant	Yhdiste, joka voi sisältää kaikkia edellä mainittuja tietotyyp-
25	DiagnosticInfo	Struktuuri, joka sisältää yksityiskohtaiset virhe ja diagnosointitiedot liittyen StatusCode-tietotyyppiin

LIITE B: VERTAILTAVIEN PROTOKOLLIEN SANOMATYYPIT

Taulukko 1. *AMQP:n kehysrunkojen tyypit. Muokattu lähteestä [52].*

Tyyppi	Yhteys	Istunto	Linkki	Kuvaus
OPEN	Käsittely			Avaa yhteyden
BEGIN	Tarkastus	Käsittely		Aloittaa istunnon
ATTACH		Tarkastus	Käsittely	Liittää linkin
FLOW		Tarkastus	Käsittely	Päivittää tiedonsiirron tilan
TRANSFER		Tarkastus	Käsittely	Lähettää sanoman
DISPOSITION		Tarkastus	Käsittely	Päivittää siirron tilan
DETACH		Tarkastus	Käsittely	Irrottaa linkin
END	Tarkastus	Käsittely		Päättää istunnon
CLOSE	Käsittely			Yhteyden sulkeminen

Taulukko 2. *HTTP-protokollan metodit. Muokattu lähteestä [19].*

Metodi	Kuvaus
GET	Käytetään palauttamaan palvelimen resurssi.
POST	Käytetään tiedon (esimerkiksi web-lomakekenttien sisällön) lähettämiseen palvelimelle, jonka palvelimen pitää käsitellä.
HEAD	Käytetään resurssin otsikon palauttamiseen ilman resurssin sisältöä.
OPTIONS	Käytetään saamaan tietoa palvelimen tai tietyn resurssin kommunikointivaihtoehdoista.
PUT	Käytetään resurssia kuvaavan uuden tiedon lähettämiseen. Käytetään joskus tiedon lähettämiseen, jos POST-metodin käyttö ei ole suotavaa.
DELETE	Käytetään resurssin poistamiseen.
TRACE	Käytetään pyynnön palauttamiseksi takaisin asiakkaalle diagnostiikkasyistä.
CONNECT	Käytetään tunnelin muodostamiseen. Tyypillisesti vain välityspalvelimien käyttämä metodi.
PATCH	Käytetään resurssin osien päivittämiseen. Ei käytössä laajasti paitsi RESTful ohjelmointirajapinnoissa.

Taulukko 3. *MQTT-protokollan ohjaussanomatyypit. Muokattu lähteestä [72].*

Nimi	**Arvo**	**Lähetyssuunta**	**Kuvaus**
Varattu	0	Kielletty	Varattu
CONNECT	1	Asiakkaalta palvelimelle	Yhteyden avauspyyntö
CONNACK	2	Palvelimelta asiakkaalle	Yhteyden avauksen hyväksyminen
PUBLISH	3	Asiakkaalta palvelimelle tai palvelimelta asiakkaalle	Sanoman julkaisu
PUBACK	4	Asiakkaalta palvelimelle tai palvelimelta asiakkaalle	Julkaisun kuittaus (QoS 1)
PUBREC	5	Asiakkaalta palvelimelle tai palvelimelta asiakkaalle	Julkaisu vastaanotettu (QoS 2 toimituksen osa 1)
PUBREL	6	Asiakkaalta palvelimelle tai palvelimelta asiakkaalle	Julkaisun julkistaminen (QoS 2 toimituksen osa 2)
PUBCOMP	7	Asiakkaalta palvelimelle tai palvelimelta asiakkaalle	Julkaisun valmistuminen (QoS 2 toimituksen osa 3)
SUBSCRIBE	8	Asiakkaalta palvelimelle	Tilauspyyntö
SUBACK	9	Palvelimelta asiakkaalle	Tilauksen hyväksyminen
UNSUB-SCRIBE	10	Asiakkaalta palvelimelle	Tilauksen peruuttaminen
UNSUBACK	11	Palvelimelta asiakkaalle	Tilauksen peruuttamisen hyväksyminen
PINGREQ	12	Asiakkaalta palvelimelle	PING pyyntö
PINGRESP	13	Palvelimelta asiakkaalle	PING vastaus
DISCON-NECT	14	Asiakkaalta palvelimelle tai palvelimelta asiakkaalle	Ilmoitus yhteyden katkaisemisesta
AUTH	15	Asiakkaalta palvelimelle tai palvelimelta asiakkaalle	Autentikoinnin vaihto